THESE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité : Informatique

Ecole doctorale : Informatique, Télécommunications et Electronique de Paris

Présentée par

Mehdi LOUKIL

Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS

GESTION DE CONTEXTE POUR L'OPTIMISATION DE L'ACCES ET L'ADAPTATION DES SERVICES SUR DES ENVIRONNEMENTS HETEROGENES

Soutenue le 20 décembre 2012

devant le jury composé de :

Directeur de thèse :	
Pr. Djamal ZEGHLACHE	Professeur à TELECOM SudParis
Encadrant :	
Dr. Badii JOUABER	Maitre de conférence à TELECOM SudParis
Rapporteurs :	
Pr. Francine KRIEF	Professeur à l'université de Bordeaux 1
Pr. Serge GARLATTI	Professeur à TELECOM Bretagne
Examinateurs	
Pr. Véronique VEQUE	Professeur à SUPELEC
Dr. Houda LABIOD	Maître de Conférences à TELECOM ParisTech
Dr. Lila BOUKHATEM	Maître de Conférences à Université Paris-Sud XI
Pr. Guy PUJOLLE	Professeur à 'Université de Paris 6 "Pierre et Marie Curie"

Thèse n° 2012TELE0055

Gestion de contexte pour l'optimisation de l'accès et l'adaptation des services sur des environnements hétérogènes

Mehdi LOUKIL

Résumé

Dans le domaine des TIC, les services de demain seront certainement basés sur des systèmes ubiquitaires, omniprésents et pervasifs. Ces systèmes devront prendre en considération différents paramètres provenant de l'environnement de l'utilisateur, c'est à dire son contexte.

Le contexte de l'utilisateur peut être composé d'informations statiques ou dynamiques, objectives ou subjectives, quantitatives ou qualitatives. Il peut inclure des données telles que la localisation géographique, les caractéristiques du terminal utilisé, la température ambiante, l'humeur de l'utilisateur.

Afin d'améliorer la QoS et la QoE, les services et les systèmes doivent être adaptés aux changements du contexte des utilisateurs. Le contexte doit donc être collecté et interprété et les règles d'adaptation du système doivent être définies. Sur les systèmes étendus, riches, dynamiques et hétérogènes, tels que ceux considéré dans le cadre de cette thèse, ces opérations doivent être automatisées.

Vu la quantité et la complexité des données contextuelles à considérer, l'utilisation de la sémantique dans la gestion de contexte peut faciliter cette automatisation et ouvrir la porte au raisonnement et à l'adaptation automatiques.

Aujourd'hui, peu de solutions viables existent pour cette problématique. Nous proposons alors d'utiliser et d'adapter des mécanismes et technologies provenant du web sémantique pour décrire et manipuler les informations de contexte. Dans un premier temps, nous avons proposé une méthodologie de conception qui nous permit de proposer « Ubiquity-Ont » : une ontologie générique au domaine des TIC, flexible et extensible. Les données de contexte ont alors été décrites sous forme de concepts et d'instances, reliés par des relations sémantiques.

Nous avons ensuite proposé une architecture overlay, composée de deux niveaux de vitalisation et permettant d'intégrer un gestionnaire de contexte, basé sur la sémantique, sur des environnements réseaux et services.

Cette solution overlay permet de (a) masquer l'hétérogénéité des composants du système et (b) d'augmenter virtuellement les entités du système existant par les capacités nécessaires à la manipulation et au raisonnement sur les données sémantiques du contexte.

Nos propositions ont étés implémentées et testées sur une plateforme réelle et appliquées à deux cas d'études : Gestion de la mobilité sur des environnements de réseaux d'accès hétérogènes et Optimisation de la consommation d'énergie dans les terminaux mobiles

Abstract

Future Information and Telecommunication Systems are expected to be pervasive and ubiquitous solutions, able to consider users' context and to automatically adapt to their environments.

Traditional configuration and management tools are not adapted. The richness, the heterogeneity and the complexity of the upcoming systems require automated solutions able to gather contextual information, to reason on them and to make the appropriate adaptation decisions. The representation and the sharing of contextual information is a key issue.

In this thesis, we proposed and used a methodology to conceive « Ubiquity-Ont », a generic ontology dedicated to Information and Telecommunication Systems. Contextual information are the described through semantic concepts, instances and relations.

We then proposed an overlay architecture, composed of two virtualization layers that can integrate a semantic context management framework over existing networking environments. This architecture is able (a) to hide any heterogeneity among the system components and (b) to augment the different entities with additional capacities for context gathering, reasoning and sharing operations.

The proposed solutions were then implemented and tested in Lab for two applications. The fisrt is related to mobility management over heterogeneous Wireless Networks and the second aims to power optimization on mobile terminals. These two case studies helped in proving and enhancing the proposed solutions.

Remerciements

Je tiens à exprimer tout d'abord mes remerciements aux membres du jury, qui ont accepté d'évaluer mon travail de thèse : Pr. Francine KRIEF, Pr. Serge GARLATTI, Pr. Véronique VEQUE, Dr. Houda LABIOD, Dr. Lila BOUKHATEM et Pr. Guy PUJOLLE.

Je tiens à remercier Badii JOUABER pour avoir accepté de m'encadrer dans cette thèse et dont l'aide précieuse m'a été indispensable sur les plans scientifiques et organisationnels. Je tiens également à le remercier pour sa confiance et ses grandes qualités humaines qui ont permis de mener à bien cette thèse. Merci à Djamal ZEGHLACHE pour avoir accepté de diriger cette thèse et pour m'avoir accueilli au sein du département RS2M depuis mon stage de fin des études. Sa confiance, son soutien et la sympathie qu'il m'a toujours témoignés ont été un facteur déterminant pour le bon déroulement de ces quatre longues dernières années.

Merci également aux différents partenaires des deux projets de recherches ANR/SEAMLESS et FUI/Wonderville, pour leur professionnalisme et pour leurs qualités humaines.

Finalement j'adresse un grand merci à toute ma famille qui a toujours été présente lorsque j'en ai eu besoin, en particulier à mes sœurs Imen & Nour, à mon frère Aymen, à mon cher père Mustapha et à ma chère mère Sabeh. Ils ont vécu à mes côtés les pires et les meilleurs moments de cette thèse.

Je tiens enfin à remercier les amis, thésards ou non qui m'ont aidé : Mariem, Takoua, Anis, Aymen, Indira, et j'en oublie certainement.

Table des matières

Table des Illustrations

Acronymes

3G : la 3ème génération de normes de téléphonie mobile
AGPL : GNU Affero General Public License
ANR : Agence Nationale de Recherche
API : Application Programming Interface
APN : Acces Point Name
BER : Bit Error Rate
BER : Bit Error Rate
BTS : Base Transceiver Station
BW : Bandwidh
CC/PP : Composite Capabilities/Preferences Profiles
CDF : Component Description Framework
COPS : Common Open Policy Service
C-OWL : Context OWL
CPU : Central Processing Unit
DL : Description Logic
ECO-WISM : Energy Consumption Optimization for Wireless Interface Selection Mechanism
EDGE : Enhanced Data Rates for GSM Evolution
FUI
FUI : Fond Unique Interministériel
GB : Gigabyte
GNU : un système d'exploitation libre lancé en 1984
GPS : Global Positioning System
GSM : Global System for Mobile Communications
IA : Intelligence Artificielle
IC : Ingénierie de Connaissance
ID : IDentifier
IDLE : désigne un processeur inactif
IMS : IP Multimedia Subsystem
IO : Input Output
IP : Internet Protocol
J : Joules
JVM : Java Virtual Machine
KB : Kilobyte
KIF : Knowledge Interchange Format
KM : Knowledge Machine
MAC : Media access control, couche du modèle OSI
MB : Megabyte
MN : Mobile Node
mSCTP : mobile SCTP
MTCP : Mobile TCP

NGN : Next Generation Network
OSI : Open Systems Interconnection, modèle d'interconnexion en réseau des systèmes ouverts
OWL : Ontology Web Language
OWL DL : OWL Description Logic
PDA : Personal Digital Assistant
PDP : Policy Decision Points
PEP : Policy Enforcement Points
PHP : PHP: Hypertext Preprocessor, langage de scripts libre.
PHY : couche Physique du modèle OSI
QoE : Quality of Experience
QoS : Quality of Service
RATP : Régie Autonome des Transports Parisiens
RDF : Resource Description Framework
RDFS : RDF Schema
RIF : Rule Interchange Format
RuleML : Rule Markup Language
SA : Scanning Association, des phases de connexions
SACMF : SEAMLESS service Adaptation and equipment Configuration Management Framework
SCMF : SEAMLESS Context Management Framework
SCTP : Stream Control Transmission Protocol
SEAMLESS : SEamless and Adaptive Services over MultipLe AccEsS NetworkS
SMMF : SEAMLESS Mobility Management Framework
SPARQL : un acronyme récursif de SPARQL Protocol and RDF Query Language
SSID : Service Set Identifier
SWRL : Semantic Web Rule Language
TCP : Transmission Control Protocol
TIC : Technologies de l'Information et de la Communication
UDP : User Datagram Protocol
UML : Unified Modeling Language
UMTS : Universal Mobile Telecommunications System
URI : Uniform Resource Identifier
URL : *Uniform Resource Locator*
W3C : WWW Consortium
WiFi : Wireless Fidelity, la norme des réseaux locaux sans fil
WURFL : Wireless Universal Resource FiLe,
XML : Extensible Markup Language

Chapitre 1 : Introduction Générale

"In the struggle for survival, the fittest win out at the experience of their rivals because they succeed in adapting themselves best in their environment". Charles Darwin, The Origine of Species.

I. Introduction

La convergence récente de l'informatique ambiante et des systèmes sensibles au contexte a généré un intérêt important pour les applications qui exploitent des aspects de l'environnement contextuel. Pour améliorer l'interaction implicite de l'utilisateur avec son environnement il faut fournir des services adaptables, présenter de l'information pertinente et gérer des mécanismes d'adaptation.

La problématique de l'informatique ambiante implique la collaboration de dispositifs hétérogènes. Ces derniers doivent agir en fonction de leur environnement et de celui de leurs utilisateurs. Il existe des éléments d'information que l'on peut considérer comme transversaux à tous les services. C'est particulièrement le cas des éléments relatifs à l'état de l'environnement physique dans lequel les services, les équipements et les réseaux sont situés ; Mais aussi le cas des éléments relatifs à la localisation, l'état de leurs ressources et leurs exigences qui peuvent varier en fonction du temps. Ces éléments font partie du contexte dans lequel les applications et/ou les services sont invoqués.

Le contexte est souvent dynamique et varie au cours du temps et en fonction d'autres paramètres et interactions. Le challenge dans de tels systèmes est de pouvoir suivre ces changements.

Notre travail consiste à concevoir et à contribuer au développement d'un système ouvert, extensible et qui peut fournir les capacités nécessaires pour développer des services et des applications consommant les informations contextuelles.

Notre hypothèse principale est de considérer un environnement totalement hétérogène. Toutefois, concevoir un système avec tous les problèmes engendrés par cette hétérogénéité, qui concerne plusieurs volets, est une tâche consistante et difficile. En effet, pour faire communiquer plusieurs machines, il faut avoir un langage commun qui unifie les concepts.

Pour ce faire, nous aurons besoin d'une approche d'interopérabilité entre softwares et d'un modèle universel pour décrire les données de contexte. Cette approche consiste à définir des interfaces de communication universelles pour masquer l'hétérogénéité des couches logicielles.

Les détails de nos réflexions, de nos propositions ainsi que nos contributions seront présentés dans les sections suivantes.

II. Cadre général et Motivation

Nos travaux de thèse s'inscrivent dans le domaine de l'intelligence ambiante. À l'origine, l'informatique ubiquitaire[1] a été proposée par Weiser [1]. D'après ses travaux, les ordinateurs deviennent invisibles dans notre environnement et sont intégrés dans les objets de la vie courante. Hier, les utilisateurs partageaient le même ordinateur, la même ressource informatique, aujourd'hui déjà, l'utilisateur dispose de son ordinateur personnel. De plus, chacun d'entre nous est bien souvent équipé au moins d'un dispositif (souvent portable) ayant des capacités de communication avec un ou plusieurs réseaux et des capacités de calcul. L'informatique se disperse dans l'environnement, elle va devenir invisible jusqu'à n'apparaître qu'en arrière-plan de notre conscience [1]. Cette orientation a donné naissance à plusieurs interprétations, qui constituent aujourd'hui plusieurs courants de l'informatique : l'informatique sensible au contexte[2], l'informatique tangible[3], l'informatique diffuse[4] et l'intelligence ambiante[5].

Les approches usuelles utilisées dans le monde des réseaux de télécommunication concentrent l'intelligence sur les équipements terminaux, tels que les routeurs d'accès, les terminaux utilisateurs, les BTS (Base Transceiver Station) et les passerelles. Les cœurs de réseaux se limitent à offrir des services Best Effort (systèmes agnostiques).

Ces approches se justifiaient jusqu'ici par plusieurs facteurs qui comprennent :

- coûts élevés et capacités limitées des équipements réseaux ;

- indisponibilité de certaines connaissances nécessaires à de meilleurs traitements et manque de mécanismes intelligents.

[1] Ubiquitous computing
[2] Context-aware Computing
[3] Tangible Computing
[4] Pervasive Computing
[5] Ambient Intelligence

Aujourd'hui, ces approches deviennent limitatives pour plusieurs raisons liées aux évolutions technologiques des réseaux et des services.

D'un côté, et avec la prolifération des applications, les besoins en termes d'adaptation et de services réseaux différenciés sont de plus en plus ressentis. Ceux-ci nécessitent de plus en plus d'intelligence malgré l'évolution des débits réseaux. De plus, l'intelligence est requise à l'intérieur des réseaux et non plus uniquement sur leur périphérie. Parmi les exemples de ces besoins, on peut citer les architectures IMS et NGN où les services d'adaptation et de différenciation qui sont nécessaires pour des services multimédia data, voix et vidéo.

D'un autre côté, la multiplication des technologies et des réseaux disponibles rend plus riche le paysage des télécom. L'interconnexion et la coopération des réseaux nécessitent des mécanismes de décision qui doivent se baser sur la connaissance d'informations contextuelles relatives à plusieurs aspects liés aux conditions physiques des réseaux, aux besoins des services et applications et même aux conditions tarifaires.

Vu la richesse de l'environnement et la complexité qui en découle pour prendre des décisions, l'automatisation semble nécessaire. Dans ce contexte, quels sont les outils et les moyens que l'on peut utiliser ? Quels sont les besoins en termes d'architectures, de raisonnement et d'échange ?

Dans cette thèse, nous proposons une approche basée sur l'utilisation des annotations sémantiques dans la description des informations de contexte. Nous proposons ensuite un Framework permettant la collecte, l'échange et le raisonnement sur ces informations de contexte et capable de prendre des décisions aussi bien sur les services réseaux (au niveau de l'accès et du routage) que sur l'adaptation du contenu.

Les mécanismes proposés sont assez génériques et flexibles pour pouvoir être étendus à différents autres aspects des réseaux et des services de télécommunication. Ils ont été déployés sur des plateformes de projets en labo pour gérer différents aspects liés à la gestion de la mobilité, de la QoS et de la réduction de la consommation d'énergie dans les réseaux.

Ainsi, un réseau bénéficiant de tels mécanismes ne peut plus être vu comme un simple système de transport de paquets, mais plutôt comme un système intelligent capable de s'auto-adapter en fonction des besoins de chaque flux.

III. Organisation et contributions de la thèse

Le présent manuscrit est composé de cinq chapitres.

Le premier chapitre analyse la gestion de contexte dans les Technologies de l'Information et de la Communication (TIC) et dans les réseaux de télécommunications en particulier. On y souligne les principaux défis et contraintes des systèmes basés sur la gestion sémantique du contexte.

Le second chapitre définit et présente l'état de l'art sur la gestion du contexte en général et dans le domaine des TIC et les réseaux de télécommunications en particulier. Il définit également la terminologie utilisée dans cette thèse et présente l'état de l'art sur les architectures et les technologies de gestion de contexte.

Le troisième chapitre est consacré à la description de la première contribution technique qui est la définition d'une ontologie pour la représentation sémantique du contexte pour les réseaux et services. Il est composé de trois parties. La première est consacrée à l'introduction de la sémantique dans la description du contexte. Elle prépare ainsi à la seconde partie de ce chapitre dans laquelle nous analysons l'état de l'art sur les ontologies. Enfin, c'est dans la troisième partie de ce chapitre que nous proposons une ontologie spécifique et détaillée tout en décrivant la méthodologie utilisée pour la construire. Ce chapitre se termine par l'évaluation de l'ontologie proposée par des tests de performances effectués sur une plateforme développée dans ce but.

Le quatrième chapitre est consacré à la présentation d'une nouvelle architecture que nous proposons pour les systèmes ambiants basés ou utilisant une gestion sémantique du contexte. L'architecture est présentée sous forme de spécifications fonctionnelles où les différents composants sont détaillés. Outre l'utilisation d'une ontologie, la solution se base sur des moteurs de raisonnement et d'inférence capables d'analyser les informations contextuelles et de prendre les décisions les plus adaptées. L'architecture proposée est définie de façon à permettre une interopérabilité optimale dans un environnement hétérogène grâce à la notion d'abstraction ou de virtualisation, par l'introduction de la notion d'*Avatars*. Les *Avatars* sont définis comme des composants logiciels capables de représenter les entités physiques au sein de la couche d'abstraction. Les performances de l'architecture proposée sont également optimisées grâce à une approche distribuée.

Les solutions proposées dans les chapitres précédents, à savoir l'ontologie et l'architecture de virtualisation, ont été mises en place et testées sur deux problématiques choisies et décrites dans

le cinquième chapitre. La première concerne la gestion de la mobilité sur des réseaux d'accès radio hétérogènes. Elle permet de récolter les informations contextuelles sur un environnement réseau étendu, riche et complexe et de prendre des décisions de handover et d'adaptation de service pour un accès sans coupure. La seconde problématique concerne l'optimisation de la consommation d'énergie sur les terminaux en les aidant, grâce aux informations contextuelles, à mieux configurer et utiliser leurs interfaces réseaux en fonction des caractéristiques de la session de services en cours.

Le manuscrit se termine par le chapitre conclusion qui résume et analyse les travaux réalisés et discute des perspectives de recherche qui restent ouvertes.

Chapitre 2 : La Gestion de Contexte dans les TIC

I. Introduction

Le contexte d'un objet est constitué de l'ensemble des connaissances relatives à l'objet et permettant de décrire son état ainsi que l'état de l'environnement dans lequel il évolue. Il permet d'analyser l'objet considéré, qu'il soit physique, logique, ou virtuel, tout en tenant compte des interactions et influences possibles qu'il peut avoir avec ce qui l'entoure.

Etudier un objet dans son contexte, permet d'avoir une vision plus complète et de mieux comprendre son comportement (si l'objet existe) ou de mieux le concevoir (si l'objet est en cours de conception).

Ce concept est utilisé aussi bien dans les sciences humaines que dans d'autres. Ainsi, dans l'analyse littéraire des textes, il est souvent fait référence au contexte d'un mot ou d'une phrase afin d'affiner sa compréhension. En linguistique, le contexte d'un mot inclut les mots qui l'entourent et permet de lui attribuer un sens en relation avec le sens global du texte. Certains mots peuvent en effet avoir plusieurs significations possibles et l'analyse de leur contexte permet d'en fixer une, la plus juste et à défaut la plus probable. Par exemple, le mot « tonneaux » sera interprété différemment dans les deux phrases suivantes et ceci grâce à l'analyse du mot et de ses interactions avec les autres mots de la phrase.

- Dans le village, l'eau de pluie est récupérée et conservée dans des tonneaux.
- Après avoir heurté un arbre, la voiture du chauffard a fait plusieurs tonneaux.

C'est donc le sens global de la phrase qui précise le sens du mot qui la compose. Alors qu'au même temps, le sens de la phrase est déduit du sens des mots qui la composent.

En histoire et en archéologie, l'analyse des événements et des faits qui nous sont transmis oralement ou par écrits, ou les objets trouvés lors de fouilles archéologiques passe souvent par l'étude des contextes « historiques », « politiques » ou « économiques ». Ces analyses permettent de placer l'évènement dans son contexte, c'est à dire de le situer de façon plus

précise dans le temps ou de déterminer le degré de fiabilité de l'histoire ou de l'événement relaté. L'étude du contexte permet ainsi de reconstituer une sorte de puzzle offrant une vue et une analyse plus globales, plus vraisemblables ou plus précises.

Aujourd'hui, la notion de contexte est utilisée dans la quasi totalité des disciplines. Il est désormais admis que pour atteindre de meilleures analyses, il est nécessaire d'englober la notion de contexte.

Qu'en est-il de la notion de contexte dans le domaine des TIC ?

Ce chapitre nous aidera à y voir plus clair. Il présente une revue de l'état de l'art sur la gestion du contexte dans le cadre des TIC. On y détaille ce que l'on entend par contexte et information de contexte, les moyens utilisés pour représenter ces informations et les outils et techniques existants qui servent à les gérer et à les exploiter. Nous présenterons aussi les principales insuffisances et les pistes possibles pour remédier à ces manques.

Ce premier travail d'investigation et d'analyse nous guidera tout au long des chapitres suivants pour affiner la problématique considérée, comprendre et assimiler l'état de l'art et apporter les meilleures solutions pour la gestion de contexte dans les réseaux de télécommunications. Nous insisterons en particulier, sur les apports de ce concept pour les systèmes de décisions dans le domaine des réseaux et services de télécommunications.

II. La notion de contexte : définition et terminologie

Les informations quantitatives et qualitatives qui caractérisent un environnement donné, ses acteurs ainsi que leurs interactions, sont primordiales pour comprendre, analyser ou prédire le fonctionnement ou le comportement d'un système. Nous appelons l'ensemble de ces informations par informations de contexte.

La détention et la maitrise de ces informations, sur lesquelles nous reviendrons plus tard dans ce chapitre, peuvent permettre également au système de s'autogérer par lui-même puisqu'elles peuvent faciliter l'auto-adaptation grâce à l'introduction possible de systèmes et de mécanismes automatisés pour la décision.

Dès 2001, Abowd a défini le contexte comme étant « *l'ensemble des informations caractérisant (partiellement) la situation d'une entité particulière* » [2].

Dourish et Chalmers ont ensuite introduit les notions de relativité et d'acceptabilité dans le contexte. « *C'est une notion dont l'acceptation n'est ni absolue ni universelle, mais relative à une situation* » [3], [4]. Cette situation peut être une situation physique comme la localisation spatio-temporelle d'une personne ou fonctionnelle comme la tâche en cours de réalisation.

Le contexte n'est pas contraint à contenir toute l'information caractérisant une situation. Ceci n'est sans doute pas possible. Au contraire, pour une même situation, il y a place pour plusieurs contextes. D'autre part, comme ces contextes peuvent être comparables en fonction des ensembles d'information qu'ils contiennent, un contexte peut être considéré comme plus ou moins étendu qu'un autre.

La différenciation et la hiérarchisation des contextes paraissent utiles et nécessaires. Ainsi, une application utilisatrice d'un *Contexte A* peut le quitter pour entrer dans un autre *Contexte B*. Par exemple, lorsqu'une personne entre dans un bâtiment, le contexte associé à la ville dans laquelle elle se situe doit passer en arrière-plan alors que celui concernant l'intérieur du bâtiment doit devenir principal, pour être de nouveau « oublié » lors de la sortie du bâtiment.

La richesse du contexte est en général fonction du besoin des fonctions que l'on vise. Ainsi si on souhaite implémenter un service de thermomètre, le contexte peut se limiter à la température. Si on souhaite implémenter un thermostat ou un service de climatisation, on agrégera les informations de température, d'ensoleillement et de temps pour réguler la température. Enfin, le contexte d'un service de station météo regroupera les cinq informations qui sont la température, la pression, l'ensoleillement (luminosité), l'humidité et le temps (date et heure) [5].

Dans le domaine des TIC, bien que plusieurs travaux se soient penchés sur la notion de contexte, les points de vue sous lesquels cette notion est abordée sont différents.

En informatique diffuse par exemple, on s'est beaucoup intéressé au contexte d'une application en termes des paramètres physiques qui caractérisent sa situation. Alors qu'en communication homme-machine, le contexte désigne le plus souvent la tâche que l'utilisateur désire accomplir et l'historique de l'interaction de l'utilisateur avec le système [3]. En intelligence artificielle, c'est encore différent, puisque le contexte est plutôt considéré comme l'ensemble des conditions qui font qu'une assertion soit valide ou non [6].

Mais commençons d'abord par introduire certaines notions et terminologies clés, que nous utiliserons tout au long de ce manuscrit de thèse.

- ***L'informatique diffuse*** (en anglais *pervasive computing*) est une notion qui fait référence à l'utilisation de composants et de capteurs communiquant les uns avec les autres [7]. Ces composants peuvent être intégrés à des objets de la vie quotidienne afin de faciliter les activités humaines dans l'environnement réel. L'idée est de fusionner les systèmes d'information et les objets du monde réel. La notion d'informatique diffuse est proche de celle de *l'informatique omniprésente* (en anglais ubiquitous computing).

- ***L'intelligence ambiante*** consiste à répartir le traitement de l'information sur tous les objets de l'environnement considéré. Les objets étant interconnectés par des réseaux. L'intelligence ambiante se base essentiellement sur :

    - l'informatique diffuse,
    - la notion de *réseaux ambiants* (en anglais Ambient Networks ou Ubiquitous Communication) qui offrent aux objets de l'informatique diffuse la possibilité de communiquer,
    - et sur les technologies des interfaces homme-machine qui permettent aux utilisateurs d'interagir avec ces objets [8].

- Les ***réseaux ambiants*** consistent à introduire une infrastructure basée en grande partie sur des logiciels (software). Cette notion permet d'offrir les moyens pour interconnecter les objets et les faire communiquer de façon transparente.
- Dans ce qui suit, nous définirons la notion de contexte dans le domaine des TIC et plus particulièrement pour l'informatique diffuse, l'intelligence artificielle et le web sémantique.
- Nous nous focaliserons ensuite sur la notion de contexte dans les réseaux de télécommunications.

II.1. Classification du contexte

La richesse et la diversité des informations de contexte nous pousse naturellement à les structurer en catégories afin de mieux les cerner. Dans la littérature, différentes classifications ont déjà été proposées [9], [10] (voir Figure 1).

Globalement, on peut distinguer quatre catégories de contexte :

- *Le contexte environnemental* : tel que les personnes ou les objets voisins, la luminosité, la température, le bruit ou le temps, …
- *Le contexte de l'utilisateur* tel que la localisation de l'utilisateur, l'intérêt personnel et les préférences, l'activité, l'émotion, le handicap, …
- *Le contexte informatique et réseau* tel que l'URL, la bande passante du réseau, la mémoire disponible, le système d'exploitation, …
- *Le contexte de temps* tel que l'historique des actions, l'historique des endroits, l'heure courante et la date.

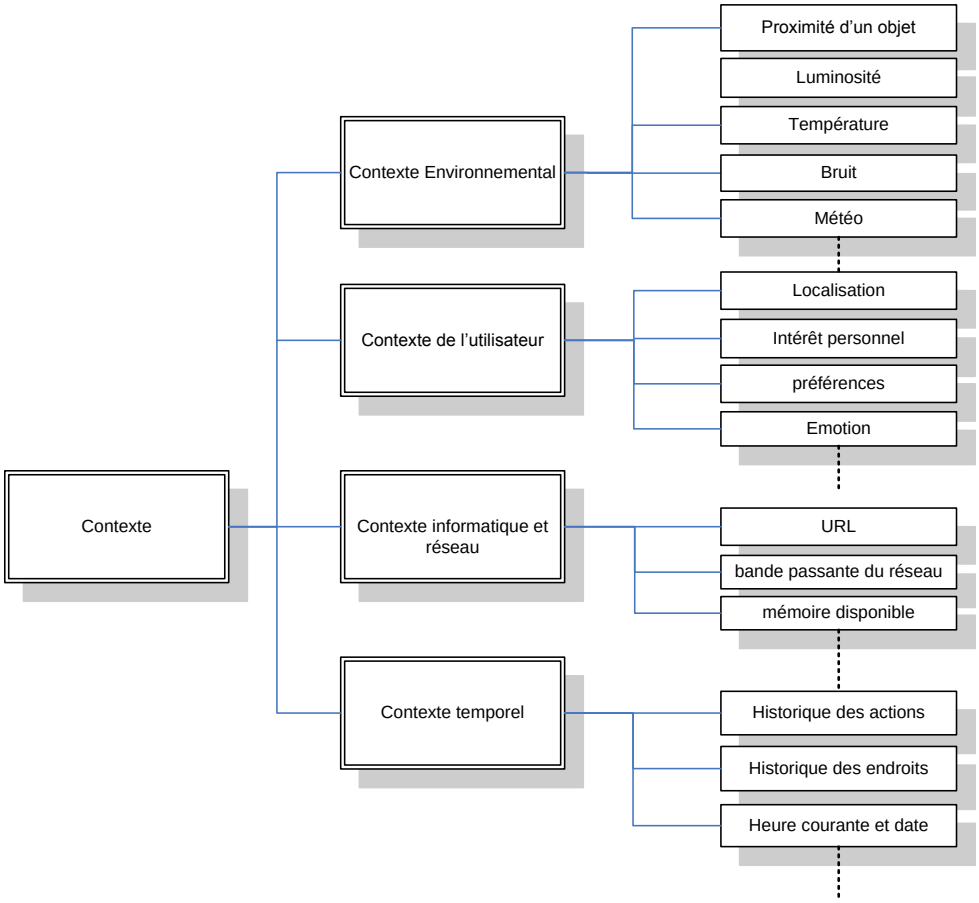


Figure 1 : Les différents types d'éléments contextuels

Nous pouvons remarquer dès à présent que ces informations ont des natures très différentes. Certaines sont objectives, facilement mesurables, quantifiables et peuvent être comparées entre elles. D'autres composantes doivent être paramétrées et numérisées comme par exemple les centres d'intérêt d'un individu, sa perception des choses, ses préférences en termes de services ou bien l'émotion qu'il ressent [11]. Certaines informations telles que les préférences devraient être saisies par les utilisateurs eux-mêmes ou bien déduites d'autres informations sur ces derniers. Enfin, certaines informations restent subjectives et difficilement quantifiables, donc difficiles à classifier ou à comparer entre elle.

II.2. Le contexte dans l'informatique diffuse

Comme on a vu précédemment, il existe des éléments d'information que l'on peut considérer comme transversaux à tous les services. C'est particulièrement le cas de la localisation spatiale, mais aussi le cas d'autres éléments relatifs à l'état de l'environnement physique dans lequel les services sont situés [6] ; [12]. Ces éléments font partie du contexte dans lequel les applications sont invoquées.

L'informatique diffuse, connue aussi sous le terme « *pervasive computing* » a pour but d'offrir des services à des utilisateurs en tenant compte de leurs interactions avec leurs environnements (y compris des objets et des êtres humains) [13] [14] et donc d'un contexte plus global. L'informatique diffuse implique la collaboration de dispositifs hétérogènes qui vont agir en fonction de leurs contextes et de celui de l'utilisateur.

Globalement, dans le domaine de l'informatique diffuse, les applications sont censées bénéficier de la connaissance du contexte dans lequel les utilisateurs évoluent, qu'il s'agisse de leurs localisations physiques, de leurs positions sociales ou hiérarchiques, de leurs tâches courantes ou des informations qui y sont liées. De plus, ces applications doivent s'adapter dynamiquement à l'irruption de nouveaux éléments dans l'environnement (e.g. d'autres utilisateurs, de nouveaux appareils ou des évènements quelconques). Ces applications ont donc besoin d'avoir en permanence la connaissance des différentes informations « utiles » caractérisant l'environnement et son évolution dans le temps.

Afin d'accéder à ces informations de contexte, l'informatique diffuse peut se baser sur divers dispositifs, certains élémentaires et d'autres sophistiqués.

Parmi les dispositifs élémentaires, nous pouvons citer deux exemples :

- les capteurs permettant de récupérer toute sorte de mesures physiques,
- les interfaces homme-machine permettant de récupérer des données sur l'état et les réactions de l'utilisateur.

Des services plus élaborés peuvent être fournis par des dispositifs actifs. Ces dispositifs génèrent à partir de données élémentaires des données plus synthétiques pouvant aller de la simple agrégation de services (comme le calcul d'une température moyenne entre toutes celles renvoyées par les capteurs) à l'application d'assistance aux utilisateurs finaux impliquant la communication avec de nombreux autres services.

Dans la suite de ce manuscrit de thèse, nous utiliserons le vocabulaire suivant :

- *L'environnement* dénotera l'ensemble des éléments caractérisant le contexte d'une application, c'est-à-dire l'environnement physique dans lequel se trouvent l'utilisateur, l'ensemble des dispositifs qui y sont présents ainsi que la description complète de la situation.
- D*ispositif,* désignera tout équipement matériel ou logiciel, présent dans l'environnement, capable de communiquer. Un *capteur* sera un dispositif sensible au contexte (context-sensitive) capable de caractériser un aspect de l'environnement physique alors que l'application sera un programme capable de fournir un service à un utilisateur.

II.3. Le contexte en intelligence artificielle

En intelligence artificielle, la notion de contexte désigne la représentation même de l'information. Elle est exploitée pour tenir compte de deux aspects conjugués :

- le contexte d'émission de l'information et plus précisément son contexte de validité
- le fait que plus on développe un raisonnement dans un contexte spécifique, plus on est efficace [6].

John McCarthy [15] a proposé une formalisation logique du contexte fondée sur une « réification » (c'est-à-dire transformer une abstraction en un objet concret) du contexte ainsi que sur le méta-prédicat ist (is true) : *ist(p,c)* signifiant que l'assertion p est vraie dans le contexte c. Les approches du contexte en intelligence artificielle permettent de grouper la connaissance en micro-théories [6] et de se placer dans ou en dehors du contexte de ces théories afin de raisonner. Dans le cadre du projet Cyc [16], le contexte permet de fournir un cadre plus précis pour l'interprétation d'une information [17].

Cette approche peut être utilisée en informatique diffuse afin d'intégrer et d'interpréter les données issues des capteurs. Ainsi, les données brutes issues des capteurs peuvent être agrégées et associées à d'autres informations permettant de les interpréter de façon plus utile. A titre d'exemple, nous pouvons citer un capteur de recul d'une voiture qui fournit 50cm comme mesure de la distance qui sépare la voiture du premier obstacle. Cette mesure est interprétée et traduite en un signal sonore qui donne indication sur cette distance. Un autre exemple, un capteur qui fournit 70% comme valeur de l'humidité dans l'air d'un endroit. Cette mesure est traduite en « le temps est humide ».

II.4. Un pont entre l'informatique diffuse et l'intelligence artificielle

Avec les deux approches précédentes, on peut considérer que l'informatique diffuse a une approche situationnelle du contexte alors que l'intelligence artificielle a une approche informationnelle. De plus, en informatique diffuse, le contexte est très souvent formé autour des besoins d'une application particulière alors qu'en intelligence artificielle, c'est plutôt la source de l'information qui en détermine le contexte. Enfin, en informatique diffuse, l'information provenant des différentes sources a tendance à être plutôt appauvrie pour être directement adaptée à l'application alors que l'intelligence artificielle tend plutôt à agréger l'information de contexte sans l'appauvrir.

Le pont entre les deux approches est effectivement d'actualité. Ainsi, dans [18], la question de l'évolution des modèles de contexte est posée. Le monde des réseaux et de l'internet pose des problématiques et des contraintes qui appartiennent à la fois au monde de l'informatique diffuse et à celui de l'intelligence artificielle. Nous nous intéresserons en particulier à cette convergence

pour la gestion du contexte dans le web dit sémantique ainsi que les services réseaux tels que la gestion de la mobilité avec la gestion de la QoS.

II.5. Le contexte dans le web sémantique

Le Web sémantique, décrit par Tim Berners-Lee et al. [19], est une extension du web classique dans lequel l'information est donnée avec un sens bien défini, afin que l'humain et les machines puissent coopérer. Une différence clé entre le Web sémantique et le Web classique réside donc dans la représentation de l'information.

En effet, dans le web classique, la représentation des données est destinée à des machines qui doivent traiter des informations au niveau de la syntaxe. Alors que dans le Web sémantique, la représentation permet aux machines de traiter et de raisonner sur l'information au niveau sémantique.

R. Guha dans [20] propose un moyen d'exprimer le contexte pour le web sémantique en associant à chaque triplet une information sur sa provenance ("quad"). Il s'agit ici d'être capable d'interpréter l'information brute provenant de sources diverses (de type fil de nouvelle). D'autres extensions sur le même modèle ont été proposées [12] et sont implémentées dans les gestionnaires de langages sémantiques.

Le langage de base du web sémantique est RDF (Resource Description Framework) [15]. Il permet d'exprimer des assertions de type sujet-prédicat-objet appelés triplets et notés <s, p, o>. C'est donc un sous-ensemble du calcul des prédicats. On peut aussi voir un tel ensemble de triplets comme un multi-graphe étiqueté et ainsi faire le lien avec le formalisme des graphes conceptuels. La force de RDF réside dans le fait que les noms des entités, qu'ils soient sujets, prédicats ou objets, sont des URI[6] (Uniform Resource Identifier, les identificateurs du web, que l'on peut voir comme une généralisation des URL : *Uniform Resource Locator*). Il est ainsi possible de faire référence à une même entité dans plusieurs documents RDF avec certitude (on peut raisonnablement supposer qu'un URI dénote la même chose quel que soit son utilisateur).

Le langage OWL (Ontology Web Language) [21], pour sa part, permet de décrire des « ontologies » ou modèles conceptuels d'un domaine particulier et rendre ainsi plus facilement

[6] http://www.w3.org/Addressing/

l'interprétation des graphes RDF concernant ce domaine. Il permet de définir des classes d'objets et de prédicats et de déclarer des contraintes pesant sur ces objets.

S'il existe des propositions comme celles de CDF (Component Description Framework) [22] ou C-OWL (Context OWL) [23], il n'existe pas à proprement dit d'outil de gestion de contexte au sens où nous l'entendons dans le web sémantique.

Mais il se trouve que les langages développés pour le web sémantique, et en particulier RDF et OWL, sont adaptés à la représentation du contexte en informatique diffuse et plus particulièrement à une représentation du contexte capable de supporter les aspects dynamiques. Ces langages ainsi que d'autres approches pour représenter sémantiquement le contexte seront présentés avec plus de détails dans le troisième chapitre de ce manuscrit.

II.6. Le contexte dans les réseaux

La notion de réseaux ambiants fait référence à des réseaux dont les mécanismes de gestion ont accès aux données de contexte (context-aware networks). Elle suppose l'introduction d'une forme d'intelligence artificielle qui augmente la capacité d'autogestion des réseaux. L'objectif est de rendre les réseaux plus autonomes et capables de s'auto-réguler en générant et en exploitant des informations contextuelles nécessaires à la prise de décisions au niveau des différentes couches du modèle [24]. La notion de contexte rejoint ainsi et enrichit les notions d'optimisation inter-couches et inter-systèmes.

Globalement, les réseaux context-aware sont supposés dépasser les réseaux intelligents en termes de fonctionnalités offertes et de performances. Ils ont été imaginés pour permettre et pour faciliter la personnalisation et la création de services tout en tenant compte de plusieurs aspects incluant les préférences de l'utilisateur, les préférences de l'opérateur du réseau ainsi que les conditions ambiantes.

Aujourd'hui, plusieurs solutions et expérimentations dans le domaine des réseaux et des services sont en cours d'étude ou en cours de spécification. Elles peuvent être considérées comme contexte-aware. Nous pouvons en citer comme exemples : le web sémantique, les Grids, les réseaux pervasifs et les réseaux autonomes (autonomic networks).

La notion de contexte dans les réseaux de télécommunications facilite la composition de nouvelles applications. Les techniques proposées pour modéliser les applications permettent d'identifier automatiquement les éléments de services qui satisfont les besoins fonctionnels et non fonctionnels [24].

Ces techniques de modélisation permettent également de décrire les applications en termes d'objectifs et de priorités. Elles permettent également d'associer à chaque application les conséquences d'un éventuel échec au moment du lancement ou d'une terminaison anormale. Ceci permet aux réseaux de gérer les possibles conflits pour l'accès aux ressources par exemple, mais surtout de gérer les applications en fonction d'une certaine « utilité ».

III. La gestion du contexte

La gestion de contexte est une nouvelle composante importante à considérer dans le développement de nouvelles applications basées sur les connaissances de leurs environnements, appelées aussi context-aware [24].

C'est une nouvelle discipline qui nécessitera certainement des spécialistes capables de définir et de gérer un cycle de vie pour le contexte. Les analystes définissent un cycle de vie composé de trois phases essentielles à savoir :

- la collecte,
- la représentation et le stockage
- la mise à disposition des informations contextuelles brutes ou synthétiques.

III.1. Collecte des informations de contexte

La méthode de l'acquisition ou de la collecte des données de contexte est très importante lors de la conception des systèmes pervasifs (sensibles au contexte). Elle dépendra mais aussi influera l'architecture du système en question. Chen [25] a défini trois façons pour la collecte du contexte: l'accès direct, l'accès via des middlewares et l'accès via un serveur dédié.

a) L'accès direct suppose l'existence de dispositifs tels que des capteurs intégrés dans les dispositifs et équipements concernés. Un logiciel client regroupe les informations de contexte directement à partir de ces capteurs plutôt que sur une infrastructure spécialisée. Dans cette approche, des pilotes (drivers) sont généralement nécessaires pour que le logiciel client puisse accéder aux capteurs embarqués. Les applications de haut niveau, peuvent avoir un contrôle sur les opérations de bas niveau de ces capteurs. Ceci a pour objectif de leur donner les moyens nécessaires pour collecter et traiter les données de contexte. Cependant, certains auteurs considèrent que cette méthode n'est pas adaptée pour les systèmes distribués, principalement pour des raisons de scalabilité. En effet, on peut facilement imaginer toute la difficulté d'accéder à un nombre important de capteurs qui peuvent en plus être hétérogènes, donc nécessitant des protocoles et des méthodes différents [26].

b) L'accès via un Middleware : L'idée ici consiste à déléguer la gestion des capteurs de bas niveau à des infrastructures intermédiaires dites Middleware. Il s'agit alors d'intégrer les outils et services composant l'infrastructure Middleware dans les équipements-mêmes ainsi que dans la plateforme qui hébergera l'application consommatrice du contexte. Dey [27] a proposé par exemple des mécanismes Middleware appelés « agrégateurs». Ces composants centralisent l'ensemble des informations de contexte relatives à une entité. On pourra ainsi, avoir un agrégateur pour chaque entité présente dans l'environnement, il collectera l'ensemble des signaux émis par les widgets concernant cet utilisateur et sera en lien avec les applications.

c) Acquérir le contexte via un serveur : contrairement aux deux premières approches, cette dernière consiste à déléguer les tâches de collecte des informations de contexte à une entité riche en ressources (ex. un serveur), différente de celle qui embarque les capteurs sources de l'information et de celle qui utilisera ou consommera le contexte. Cette approche peut à priori apporter des réponses pour remédier aux inconvénients des deux premières approches. Elle peut en effet minimiser les traitements sur les équipements terminaux, souvent limités en ressources. D'un autre côté, l'utilisation d'un serveur dédié aux informations de contexte facilite la réutilisation des capteurs par plusieurs utilisateurs [26]. En effet, les applications consommatrices des informations issues des capteurs vont pouvoir solliciter le serveur de contexte et éviter ainsi des accès bloquants aux capteurs. L'utilisation d'un serveur de contexte riche en ressources, peut également permettre de conserver l'historique de plusieurs données contextuelles récupérées au fil du temps. Ainsi il devient possible de détecter et de résoudre des éventuels conflits ou

incohérences entre les informations acquises à partir des capteurs. Cependant, l'utilisation d'un système de gestion de contexte basé sur l'architecture client-serveur doit tenir compte de plusieurs autres facteurs tels que les protocoles à mettre en place, les performances des liens réseaux et la qualité des paramètres de QoS (temps de réponse, disponibilité, redondance, …).

Les infrastructures de gestion des informations contextuelles collectent des informations de l'environnement pour les diffuser et distribuer par la suite. Elles ont le plus souvent un modèle abstrait pour les représenter afin de faciliter l'échange des informations, utilisables par les applications et services consommateurs de contexte, et assurer une compréhension unifiée de ces informations d'un composant ou dispositif à l'autre.

III.2. Représenter et stocker le contexte

Dans les sections précédentes nous avons défini le contexte dans plusieurs domaines et nous avons discuté de la nécessité de gérer le contexte en commençant par sa collecte ou son acquisition. Cependant, conserver les informations de contexte en leur état brut n'est souvent pas suffisant.

Même si généralement, un contexte concerne un environnement bien défini à un instant bien déterminé, le besoin de le stocker peut être utile pour plusieurs raisons. L'utilité du contexte peut souvent dépasser son cadre géographique et temporel.

Alors pourquoi stocker le contexte et pourquoi ne pas se limiter à une utilisation dynamique ?

Le stockage des informations contextuelles consiste souvent à mémoriser les valeurs que prennent les composantes de celui-ci au cours du temps. Ces sauvegardes présentent plusieurs intérêts :

- pour l'infrastructure de gestion de contexte elle-même : les données stockées représentent une base d'informations pour la récupération en cas de pannes,
- pour les développeurs de systèmes informatiques ambiants : le stockage permet de calculer des indicateurs et des tendances et d'alimenter un système d'apprentissage permettant dans certains cas d'affiner ou de prédire des informations manquantes.

- Enfin, le stockage du contexte peut optimiser le système en lui-même en permettant de limiter les opérations d'échange aux données dynamiques. Les données statiques étant sauvegardées une fois pour toute.

Les données de contexte peuvent représenter une mine d'informations, à condition que les formats de représentation et de sauvegarde soient adaptés. En effet, une représentation classique, sans vraie structuration, ne permettrait pas aux applications, services et systèmes d'exploiter tout le potentiel donné par la détention de telles données. Rendre ces informations compréhensibles et manipulables par des machines et des automates permettrait de raisonner dessus et d'automatiser certaines prises de décision et les adaptations nécessaires et souhaitées [2].

Dans [28], Buchholz et al. ont discuté des exigences concernant la représentation des données de contexte. Ils ont défini pour cela un certain nombre de critères à respecter. Ainsi, selon les auteurs, une bonne représentation de contexte doit être :

- structurée,
- interchangeable,
- uniforme,
- extensible,
- standardisée,
- décomposable.

Le système de gestion des données de contexte doit offrir plusieurs fonctionnalités telles que la sauvegarde, la mise à jour, la recherche, le croisement de données et la compilation des informations. L'ensemble de ses fonctionnalités doit être conçue en fonction du système et de ses besoins et exigences. Ces aspects seront traités plus en détail dans le troisième chapitre.

III.3. Le contexte et l'incertitude : le besoin d'enrichir et de compléter le contexte

La fiabilité, la complétude et l'incertitude sont des aspects importants dans la manipulation de toute donnée sensible sur laquelle doivent se baser des décisions. Comment alors mesurer le degré de fiabilité des données contextuelles ?

Dans la littérature, la définition de l'incertitude a évolué au cours du temps et selon les disciplines et les types de recherche. Nous pouvons néanmoins retenir trois principales définitions. Tout d'abord, certains auteurs utilisent la définition énoncée par Knight [29] qui conçoit l'incertitude comme étant l'incapacité d'affecter une probabilité à l'occurrence d'un évènement. L'incertitude caractérise ainsi une situation dans laquelle les acteurs ne peuvent pas prévoir ce qu'il peut et va se passer [30]; [31]). Knight distingue alors le risque de l'incertitude. Une situation risquée étant une situation pour laquelle les hommes sont capables d'affecter une probabilité à l'arrivée d'un événement. Cependant, les auteurs ne font pas toujours la différence entre les deux, ce qui a amené Herring [32] à considérer que, pour des raisons pratiques, risque et incertitude sont des synonymes. C'est de là qu'on peut parler de la deuxième définition qui considère que l'incertitude résulte de la difficulté à prévoir les évènements. Enfin, l'incertitude peut être perçue comme le manque d'informations sur un sujet, un facteur ou une situation. Dans ce cas, il est alors considéré que plus on est à même de réunir de l'information, moins on sera soumis à l'incertitude comme expliqué par Daft et Lenge [33]. La qualité de l'information obtenue est également importante selon Atkinson, Crawford et Ward [34]. Plus la qualité de l'information, sa fiabilité, les détails de celle-ci sont bons, moins il y aura d'incertitude. Galbraith, qui s'intéresse à l'incertitude d'un point de vue technologique indique que l'incertitude provient ainsi de la différence entre l'information nécessaire pour réaliser la tâche et la quantité d'informations déjà en possession de l'organisation ou du système [35].

Les systèmes de gestion de contexte doivent naturellement prendre en considération le manque d'informations contextuelles pour diverses raisons techniques, architecturales ou de performance. On peut remédier partiellement à ce manque par des systèmes de prévisions et d'extrapolations. Le système peut en effet incorporer des moteurs de raisonnement qui analysent le contexte actuel pour détecter les données manquantes ou le degré d'incertitude. Différentes techniques peuvent être utilisées telles que l'apprentissage, l'analyse comparative des contextes similaires, l'analyse de l'historique du contexte pour éventuellement prédire ou compléter les informations manquantes. D'un autre côté, un système de gestion de contexte intelligent, pourrait anticiper et prévoir les événements futurs. Ces capacités de remédier à l'incertitude et à prédire des événements futurs peuvent être très bénéfiques pour la gestion des systèmes complexes.

Dans la suite de nos travaux, et en nous inspirant des travaux de Chenhall et Morris [36], nous avons choisi de définir l'incertitude comme étant la réunion entre (a) un manque d'informations

sur une situation particulière et (b) la difficulté de prévoir les événements qui vont se produire et les conséquences qu'ils auront sur le système considéré.

III.4. Publier et fournir le contexte

Ce sont là les opérations nécessaires permettant aux applications et services de consommer les données contextuelles. Une fois récupérées, traitées et stockées, les données contextuelles doivent être mises à la disposition des applications et des services qui pourront les exploiter pour pouvoir prendre en compte la dynamicité de l'environnement.

Les informations de contexte doivent donc être accessibles ou parvenir aux différentes applications concernées ou abonnées. Une des difficultés majeures est que les flux d'informations concernant les informations de contexte ne peuvent pas être déterminés à l'avance. Ils doivent se construire à la volée, pendant l'exécution. Ceci est dû essentiellement à un certain nombre de facteurs :

- la dynamicité des données contextuelles : quels sont les éléments de contexte qui ont « suffisamment » changé et qui nécessitent de les fournir de nouveau à une application consommatrice qui en possède déjà une copie préalable ?
- les besoins de l'application consommatrice : quelles sont les données contextuelles utiles à une application donnée ? Il est évident que chaque application ou service aura besoin d'un sous ensemble de ces informations et non de sa totalité.
- Les mises à jour des applications et des services : des versions successives d'un même service ou d'une même application peuvent avoir des exigences différentes en termes de ressources. Certains services peuvent également offrir plus de classes de QoS (Quality of Service). Dans ces cas, les besoins du même service en termes de données de contexte peuvent varier dans le temps.
- Les droits d'accès aux données : certaines données peuvent être propriétaires, ayant un caractère de confidentialité ou touchant aux aspects de la vie privée. On pourrait donc imaginer que certaines données nécessitent des droits d'accès liés à la sécurité ou à la facturation.
- Les préférences et les habitudes des utilisateurs : les utilisateurs peuvent changer de préférences selon le contexte (le jour, le soir, selon l'endroit etc.). D'autre part, la

plupart des utilisateurs ont des habitudes qui se répètent, donc on pourrait imaginer ou anticiper certaines données contextuelles en fonction des habitudes sauvegardées.

D'un autre côté, la publication et la mise à disposition des données de contexte nécessitent des protocoles adaptés à la nature et aux caractéristiques des données transportées. L'introduction de la sémantique dans la description des données contextuelles, additionnée par les notions d'incertitude, de droit d'accès et de temps comme discuté plus haut, requiert des mécanismes intelligents et optimisés, dépassant la notion classique de base de données et de client-serveur.

Dans la littérature, les modèles de gestion de l'information de contexte existants traitent assez peu cet aspect dynamique [37]. Il n'existe presque pas de solutions globales traitant tous les aspects que nous venons de discuter et offrant les interfaces d'échanges d'informations de contexte entre le système de gestion et les consommateurs de ces informations.

Tous ces aspects seront considérés dans les prochains chapitres et nous guideront dans nos propositions d'une architecture globale pour la gestion des informations de contexte pour les réseaux.

IV. Architectures et approches pour faciliter la gestion du contexte

Aujourd'hui, plusieurs systèmes et services réseau qui fournissent des services sont basés en partie sur des informations contextuelles. Nous pouvons en citer les exemples les plus répandus :

- Présentation de pages web ou d'interfaces d'applications adaptées aux terminaux des utilisateurs, en tenant compte en particulier de la taille de l'écran et de la plateforme logicielle embarquée. En effet, plusieurs sites et applications offrent aujourd'hui, soit plusieurs versions soit des moulinettes d'adaptation, permettant de présenter aux utilisateurs le rendu le plus adapté à son terminal. Le système détecte certaines caractéristiques du terminal (type de navigateur, système d'exploitation, …) et adapte son interface en fonction. D'autres données contextuelles peuvent être utilisées telles que la mobilité et le débit disponible à l'accès.

- Adaptation du service au contexte et aux besoins de l'utilisateur. Plusieurs services permettent aujourd'hui de personnaliser les interfaces en fonction de certains paramètres récoltés auprès de l'utilisateur ou en se basant sur son expérience passée avec le service. Certains aspects des services, au niveau fonctionnel, peuvent être adaptés en fonction par exemple de la localisation (GPS), des niveaux d'éclairage et de bruit, et même de la présence ou non d'autres personnes dans l'entourage. Ainsi, une carte géographique avec les restaurants les plus proches va être présentée à l'utilisateur en tenant compte de la connaissance de sa localisation dans la ville et même en fonction de ses préférences gastronomiques ou de prix si celles-ci sont connues.
- Optimisations des systèmes et réseaux ; Grace aux données contextuelles, il est possible aujourd'hui d'aider le terminal utilisateur à mieux gérer sa mobilité en lui fournissant le meilleur accès possible en fonction des connaissances que l'on peut avoir concernant son environnement radio. Ce genre d'optimisation a déjà été utilisé dans les systèmes GSM (Global System for Mobile) en surveillant la liste des cellules voisines. Aujourd'hui, avec la multiplication des réseaux WiFi, certains opérateurs ont intégré des systèmes de décision permettant aux mobiles de s'attacher aux réseaux d'accès le plus adapté (GSM, 3G ou WIFi), souvent du point de vue opérateur.

Afin d'offrir ce genre de service et d'application et d'atteindre des objectifs d'optimisation inter systèmes par exemple, l'architecture de gestion de contexte choisie doit intégrer un certain nombre d'outils assez flexibles et ouverts, qui permettront aux concepteurs d'applications et aux opérateurs réseaux de rendre leurs services adaptables et personnalisables. Nous présentons dans la section suivante l'état de l'art concernant ces architectures.

IV.1. Architectures et approches existantes

Plusieurs propositions pour gérer les données contextuelles existent dans la littérature. La plus part d'entre elle cible des systèmes particuliers et n'est pas assez générique et ouverte.

Dans [38] les auteurs proposent d'utiliser le cadre CC/PP (Composite Capabilities/Preferences Profiles), pour décrire les caractéristiques des terminaux mais aussi les préférences utilisateurs.

Asadi propose [39] d'utiliser le standard MPEG-21[7] pour l'adaptation de la présentation des contenus multimédia en fonction du contexte sémantique. En effet, MPEG-21 contient lui même des outils pour la description des caractéristiques liées aux réseaux et aux terminaux : formats supportés, capacité d'affichage vidéo et audio, capacité de stockage, périphériques, etc.

Dans [40], les auteurs proposent d'utiliser le format Objet pour décrire les différents éléments du contexte. Dans l'exemple qu'ils considèrent, chaque élément du contexte physique : salle de cours, matières du cours, professeurs, étudiants, etc. est représenté par un objet. Ces objets forment une hiérarchie qui peut être manipulée et enrichie dynamiquement.

D'autres propositions se réfèrent aux modèles de référence des réseaux (modèle OSI). Le projet MAGNET (My personal Adaptive Global NET) [41] par exemple propose de gérer les données contextuelles aux niveaux des différentes couches du modèle OSI en y intégrant des capteurs (sources des informations de contexte).

Un des domaines qui a le plus bénéficié d'expérimentations pratiques sur la gestion des données de contexte est celui des réseaux de capteurs.

L'infrastructure « Context Toolkit » a été proposée et construite autour d'un modèle faiblement centralisé. C'est une boite à outils inspirée des bibliothèques graphiques offertes aux applications informatiques. L'infrastructure de gestion de contexte offre un certain nombre de fonctions centrées sur la localisation des capteurs :

- les interacteurs qui offrent un accès simple aux capteurs,
- des interpréteurs qui donnent sens aux données capturées,
- des agrégateurs qui font le lien entre les intéracteurs et les applications sensibles ou consommatrices des données de contexte.

La deuxième infrastructure est proposée par Rey [42]. C'est une infrastructure de gestion de contexte totalement décentralisée, basée sur des composants appelés contexteurs. Un contexteur est capable de s'auto-décrire et de fournir son nom, sa classe, ses interfaces IO (Input Output) ainsi que les fonctions qu'il est capable d'offrir.

[7] MPEG-21 est l'acronyme d'une norme internationale ISO/CEI développée par MPEG (Moving Picture Experts Group) dont l'ambition est de spécifier une architecture permettant l'interopérabilité et l'utilisation transparente des représentations audiovisuelles numériques (multimédia).

Les services contextuels sont structurés en une pyramide à quatre niveaux d'abstraction. L'infrastructure offre les services de base suivants :

- modéliser la transformation d'observables de phénomènes physiques (couche capture) provenant directement de l'environnement vers des observables symboliques,
- identifier les entités, les relations, les rôles et les attributs pertinents (couche transformation),
- établir des réseaux de contextes (couche identification),
- établir l'interface avec les applications au bon niveau d'abstraction (couche adaptation).

A ces services de base, s'ajoute un certain nombre de services transversaux tels que la consultation de l'historique et la découverte de services.

L'interopérabilité entre chaque contexteur est possible grâce à l'utilisation du langage XML et de schémas de données connus a priori.

IV.2. Synthèse

L'analyse de ces différentes propositions, montre qu'elles sont souvent destinées à des scénarios d'utilisation bien déterminés. Certaines ciblent plutôt la description du contenu des services (multimédia, vidéo, image, …) d'autres sont plutôt conçus du point de vue de l'opérateur réseaux (e.g. proposition MAGNET).

Une autre limitation importante qui ressort de l'analyse de ces différentes solutions concerne le traitement même des informations du contexte, qui se base le plus souvent sur des requêtes syntaxiques voire à base de règles. La plus part des solutions introduisent peu ou pas assez la notion de sémantique facilitant l'automatisation.

Plusieurs faiblesses et inconvénients ressortent de notre analyse sur les approches existantes :

- La sémantique non explicitée dans la description des contextes de l'ensemble application/service/réseau/utilisateur rend celui-ci interprétable uniquement par le développeur du service. Il n'est donc pas possible de permettre la découverte et l'interprétation de ces informations voir de ces règles par des logiciels.

- Il existe divers technologies et langages différents et spécifiques pour décrire les informations de contexte. Or l'utilisation d'une multitude de langages rend difficile l'émergence d'une intelligence globale ou d'un système universel multi-domaines.
- Le concepteur d'un service doit prévoir a priori des règles concernant tous les états pour un contexte donné. Il s'agit d'un effort important et presque surhumain surtout pour des contextes riches et donc complexes. La conception et le développement de nouveaux services deviennent presque impossibles à l'échelle humaine à cause du nombre important de règles et des comportements à prévoir.
- Les règles d'adaptation des services sont spécifiques à chaque service et ne peuvent pas toujours être généralisées.

Dans [43], Thomas Strang et Claudia Linnhoff-Popien proposent une comparaison de différentes solutions existantes pour modéliser les données de contexte :

- les modèles clef-valeur,
- les modèles à balise de type XML (Extensible Markup Language),
- les modèles graphiques de type UML (Unified Modeling Language),
- les modèles orientés objets,
- les modèles logiques comme celui de McCarthy et
- les modèles basés sur des ontologies

La comparaison de ces modèles est basée sur les six critères suivants :

- Composition distribuée,
- Validation partielle,
- Richesse et qualité de l'information,
- Non-complétude et ambiguïté,
- Niveau de formalisme,
- Applicabilité aux environnements existants

Les auteurs concluent que les modèles basés sur les ontologies se révèlent les plus adaptés à la modélisation du contexte dans les environnements d'intelligence ambiante. De plus, d'après eux, les ontologies sont particulièrement adaptées pour décrire les éléments d'information relatifs à notre vie quotidienne dans une structure de données utilisables par des ordinateurs.

D'un autre côté, et concernant les architectures cette fois, on peut parler de trois approches possibles : fortement centralisée, faiblement centralisée et décentralisée.

- Dans l'approche fortement centralisée, un composant central (i.e. un serveur de contexte), prend en charge l'ensemble des services liés à la gestion du contexte. Autour de ce serveur, une collection de composants indépendants (les clients et les producteurs) opère pour produire ou consommer les informations de contexte. La centralisation signifie que le serveur est le point de passage obligé de toute transaction entre les producteurs d'information (tels les capteurs) et les consommateurs. Cette centralisation n'exclut pas qu'au niveau de la structure physique, le serveur soit réparti sur plusieurs machines.
- Dans l'approche faiblement centralisée, un composant central prend en charge la tâche de découverte de services. Il met en relation le client (application) et la source du service (par exemple un composant de capture). Les autres fonctions de l'infrastructure de contexte sont attribuées à d'autres composants spécifiques répertoriés dans le composant central. Contrairement au cas précédent, ici le serveur n'est qu'un annuaire de services. Il n'est plus le point de passage obligatoire de toutes les transactions entre les producteurs et les clients mais seulement des messages d'enregistrements et de découvertes.
- Dans l'approche décentralisée, il n'y a pas de composant central. Les services sont alors répartis dans un ensemble de composants. Lors d'une recherche, les messages transitent de composant à composant jusqu'à atteindre leurs destinataires. Une fois le fournisseur du service trouvé, la communication fournisseur-client peut alors se faire directement. Comme pour les infrastructures faiblement centralisées, les systèmes décentralisés utilisent principalement un modèle en flots de données où les entrées des composants clients sont les sorties des composants producteurs.

Chacune de ces trois approches possède des avantages et des inconvénients. Bien que les infrastructures fortement centralisées soient plus facilement administrables, elles nécessitent l'utilisation de machines puissantes, capables d'assurer le rôle du serveur. Elles sont vulnérables et peuvent être entièrement paralysées en cas de panne du serveur. Pour sa part, le modèle faiblement centralisé limite les fonctions du serveur, ce qui a pour effet de diminuer la puissance nécessaire de la machine serveur et de faciliter la mise en place de capacité de reconfiguration. Inversement, l'administration de l'infrastructure est plus complexe. Contrairement aux approches fortement et faiblement centralisées, le modèle décentralisé dilue

les fonctions de l'infrastructure dans chacun de ses composants. La vulnérabilité due au composant central disparaît en même temps que la nécessité de machines performantes. Une machine peut exécuter autant de composants que sa puissance le lui permet : peu sur un PDA (Personal Digital Assistant), beaucoup sur une station de travail de haute performance. Cependant, la gestion, l'administration et l'optimisation de telles architectures sont souvent très complexes.

V. Conclusion

Dans ce chapitre, nous avons présenté et discuté l'état de l'art sur la gestion de contexte dans les TIC.

Après avoir présenté la terminologie et l'origine de cette notion, issue des sciences humaines, nous avons présenté les différentes définitions que l'on peut retrouver dans la littérature. Nous remarquons qu'il existe plusieurs définitions du mot contexte, selon les secteurs d'activité de leurs auteurs. Cette notion reste donc assez subjective.

Dans le domaine des TIC, la gestion de contexte a été introduite avec l'avènement de l'intelligence artificielle, puis reprise avec les notions d'informatique diffuse et d'informatique omniprésente. L'idée est de contrer la complexité des systèmes de plus en plus étendus et hétérogènes, par l'introduction d'une certaine forme d'intelligence et d'autonomie. On vise alors des systèmes auto-gérés ou auto-adaptatifs, capables de se reconfigurer en fonction de leurs contextes.

Différents travaux ont utilisé ou traité la notion de contexte dans les TIC. L'analyse de cette littérature nous a permis de proposer une classification des données contextuelles les plus souvent évoquées dans ce domaine. Celles-ci peuvent englober des données comportementales telles que l'émotion de l'utilisateur face à une situation, environnementales telles que la luminosité ou la température, ou techniques telles que la mémoire disponible ou la technologie réseaux disponible sur le terminal de l'utilisateur.

Nous avons également montré que toutes ces informations peuvent ne pas être disponibles ou mises à jour, à un point donné ou pour un instant donné. Il y a donc des facteurs d'incertitude et complétude que certains auteurs ont introduits.

Nous avons ensuite abordé le cycle de vie de la gestion de contexte ainsi que les différentes techniques et technologies associées. Nous avons montré que le contexte doit être identifié, collecté, représenté, stocké, analysé et publié. Pour chacune de ces étapes, les principales approches présentes dans la littérature ont été présentées. Pour les opérations liées à la représentation, au stockage et à l'analyse, nous avons choisi de leur consacrer le Chapitre 3.

Enfin, nous avons également abordé les architectures et les approches, gérant ou utilisant les informations de contexte, proposées dans la littérature ou à travers divers projets. On y trouve des solutions centralisées, décentralisées ou hybrides. Ces solutions que nous avons présentés dans cet état de l'art, les infrastructures existantes offrent, pour la plupart, des fonctionnalités qui facilitent et enrichissent le service de gestion d'informations de contexte, mais aucune d'entre elles ne peut être réellement déployée dans un environnement ouvert et complexe des TIC. De par leur ouverture ces environnements sont très peu contraints et les infrastructures détaillées dans ce chapitre ne supportent généralement que partiellement les spécificités de ce type d'environnement : hétérogénéité, dynamisme et extensibilité des modèles de contexte. En effet, toutes ces solutions supposent que les entités de l'environnement considéré sont capables de générer et de communiquer leurs informations de contexte, voire de raisonner et de prendre des décisions d'adaptation en fonction.

Or, à cause de la richesse de la notion de contexte et la complexité qui peut en découler pour accéder et raisonner sur ces informations, il est certain que quelques entités seraient incapables d'accéder et de manipuler des données de contexte, spécialement sur des environnements hétérogènes. C'est à cette problématique que les prochains chapitres vont essayer d'apporter des solutions.

Chapitre 3 : Ubiquity-Ont : Ontologie Générique dans un environnement de réseaux et de services hétérogènes

Partie 1 : De la sémantique aux Ontologies

I. Introduction

Avec l'accroissement des volumes de données manipulées par les systèmes d'information, le traitement symbolique des informations apparaît comme le seul moyen viable pour les systèmes du futur. D'autant que la convergence entre des systèmes historiquement indépendants, tels que les systèmes de télécommunication, les services informatiques, les réseaux de transport, le transport d'énergie, et bien d'autres deviennent chaque jour une réalité.

Les premières tentatives de traitement automatisé de l'information peuvent être attribuées à l'Intelligence Artificielle, qui, la première, a essayé d'introduire de la sémantique dans la description des données traitées, à travers des règles. La robotique a ensuite utilisé certains outils de l'intelligence artificielle pour élaborer des machines intelligentes capables de mimer le comportement humain.

Les problématiques communes à l'intelligence artificielle et à la robotique étaient essentiellement la représentation des connaissances et des inférences ou règles aboutissant au choix de l'action la plus appropriée.

Dans les autres systèmes informatiques ou de télécommunication, les difficultés majeures étaient des problèmes d'optimisation et de configuration. La plupart des problèmes étaient résolus par des solutions d'ingénierie effectuées manuellement et on a jusqu'ici très peu cherché des solutions d'automatisation venant des domaines de l'intelligence artificielle.

Ceci s'explique essentiellement par deux facteurs :

- L'immaturité des systèmes de représentation qui étaient peu génériques et incompréhensibles voire inutilisables par les spécialistes des différents domaines.

- Le fait que les systèmes de télécommunication ou d'information, même en étant complexes technologiquement, étaient suffisamment circoncis à un sous domaine particulier, qu'un nombre de quelques ingénieurs et techniciens étaient capables de configurer, dimensionner et gérer.

Aujourd'hui, à cause des convergences verticales et horizontales qu'on observe et qu'on prévoit pour les systèmes cités plus haut, la gestion et la configuration de ceux-ci deviennent difficiles à gérer manuellement. D'autant que le nombre d'utilisateurs et de services offerts ne cesse de croire avec en plus, des ambitions de personnalisation et d'adaptation des différents services en fonction des souhaits et contraintes de chaque utilisateur.

Ainsi, les référentiels, c'est à dire l'ensemble des objets manipulés, de chaque domaine se retrouvent inter-mêlés et interdépendants. Les techniques informatiques classiques de représentation ne semblent pas adaptées à représenter et à raisonner sur de telles structures complexes. L'introduction de la sémantique pour une représentation plus intelligente semble donc une solution prometteuse.

II. La sémantique : origines et signification

La sémantique est à l'origine une branche de la linguistique où chaque phrase est composée d'un signifiant et d'un signifié. Par signifié, on désigne ce dont on parle et par signifiant, on désigne la syntaxe utilisée pour parler du signifié.

Dans la linguistique, la sémantique possède plusieurs objets d'étude dont :

- la signification des mots et des mots composés,
- les rapports de sens entre les mots (relations d'homonymie, de synonymie, d'antonymie, de polysémie, d'hyperonymie, d'hyponymie, etc.),
- la distribution des acteurs au sein d'un énoncé,
- les conditions de vérité d'un énoncé.

En informatique, le terme de sémantique est utilisé en opposition à celui de syntaxe. Les deux vont de pair pour décrire les langages de programmation informatique (e.g. C, C++ JAVA, …), la sémantique décrivant le fond et la syntaxe décrivant la forme.

On a également cherché à enrichir les données par de la sémantique, en commençant par choisir des représentations plus parlantes. Ainsi en traitement de données, les méthodes de fouille de données (en anglais *data mining*) permettent d'analyser un ensemble de données à priori disparates, en dégageant des classifications avec formes issues de l'intelligence artificielle :

- Les tableaux
- Les graphes (réseau maillé d'objets, de concepts, etc.) et les arbres

Cependant, ces représentations restent des signifiants, au sens où elles représentent les données. Mais leur appartenance à une lignes/colonne d'un tableau, à une branche d'un arbre ou d'un graphe apporte de la sémantique relative à l'appartenance de la donnée à un groupe donné, en fonction des critères de classification utilisés. L'analyse des données peut donc être plus aisée grâce à cette sémantique.

Depuis quelques années, des langages formels dédiés à la description sémantique sont apparus et on parle de plus en plus d'ontologies. Nous allons décrire ces aspects dans les sections suivantes.

III. La notion d'Ontologie

Une Ontologie peut être définie comme étant une spécification formelle explicite des termes et des relations entre les termes d'un domaine particulier [44]. Cette notion, initialement développée dans les laboratoires d'Intelligence Artificielle, est de plus en plus utilisée aujourd'hui dans les domaines des TIC. En effet, l'utilisation des ontologies est devenue très courantes dans le World-Wide Web. Le champ d'application des ontologies dans le web est très large, allant de la catégorisation des sites Web (tels que dans Yahoo!) à la catégorisation des produits destinés à la vente et de leurs caractéristiques (tel que dans Amazon.com). Dans le domaine du web, le W3C (WWW Consortium) a développé un langage de description de la connaissance sur les pages Web, afin de faciliter la recherche

électronique automatisée. Il s'agit du langage RFD (Resource Description Framework) [45].

Dans la littérature, différentes raisons sont évoquées pour défendre et promouvoir l'utilisation des ontologies dans les domaines des TIC.

- Partager une compréhension commune de la structure de l'information entre les fabricants de logiciels : est une des raisons les plus courantes qui conduisent à développer des ontologies [44] [46]. Un des exemples les plus cités est celui d'un ensemble de sites web fournissant de l'information médicale ou des services de e-commerce en médecine. La définition et l'utilisation d'une ontologie médicale peut très bien faciliter l'extraction automatisée de données croisées à partir des différents sites et peut même favoriser la création de nouvelles applications de médecine utilisant ces sites.
- Permettre la réutilisation du savoir sur un domaine : Un des exemples les plus simples est la représentation du temps. Certaines applications ont besoin du temps absolu, d'autres se suffisent d'un temps relatif. Aussi, le temps est généralement défini par des intervalles tels que les secondes et les minutes. Une ontologie spécifique mais générique au domaine du temps peut être développée et réutilisée par plusieurs domaines d'applications, chacune de ces dernières pouvant l'instancier et l'enrichir ou l'étendre à sa manière.
- Expliciter ce qui est considéré comme implicite : Il est évident que les postulats implicites utilisés dans certains domaines et exprimés en langage codé de programmation sont difficiles à comprendre en dehors des personnes qui les ont programmés. S'il est possible de rendre explicites les postulats d'un domaine, la coopération entre personnes et la réutilisation des codes seront plus aisées.
- Séparer le savoir sur un domaine du savoir opérationnel : les opérations de configuration et de maintenance d'un domaine (e.g. un réseau) peuvent être facilitées si ces mêmes opérations peuvent être plus facilement déduites de ses constituants. Ceci implique que les spécifications des constituants doivent décrire d'une façon claire et séparée les fonctionnalités du domaine et les fonctionnalités permettant d'opérer et de configurer ce domaine [47]. Ainsi, l'on pourra imaginer des outils génériques basés sur des ontologies, et permettant de s'auto-adapter pour configurer tel ou tel équipement d'un domaine. Ces algorithmes appliqueront alors les mêmes logiques sur des ontologies similaires [48].

- Analyser le savoir sur un domaine : Avec l'utilisation des ontologies, l'analyse formelle, extrêmement précieuse, devient possible et plus accessible sur différents domaines. Il devient alors plus facile de les analyser, de les comprendre, de les utiliser et de les étendre [49].

III.1. Qu'est une ontologie ?

Le mot ontologie tire son origine de la philosophie. Il a ensuite trouvé une application dans diverses disciplines [50]. Aujourd'hui, il existe plusieurs interprétations de ce mot selon les domaines.

Les premières divergences sont apparues entre la pensée philosophique et l'approche de l'Intelligence Artificielle (IA) dans les systèmes d'information. D'autres confusions dans la définition et l'application de l'ontologie existent dans les distinctions (pas claires) entre ontologie (au singulier) et ontologies (au pluriel), ontologie et base de connaissance, et ontologie et épistémologie.

En logique, le quantificateur existentiel ∃ est une notation pour évaluer que quelque chose existe, mais la logique elle-même n'a pas de vocabulaire pour décrire les choses qui existent. Les ontologies remplissent cet espace, et elles sont utilisées pour étudier l'existence de toutes sortes d'entités, abstraites et concrètes, qui composent le monde [SOWA00]. Dans le dictionnaire anglais d'Oxford [OED03], l'ontologie est définie comme appartenant au "domaine métaphysique" et comme la "science ou l'étude de l'existant". Selon Wordnet [WORD98], l'ontologie peut être définie comme "l'étude métaphysique de la nature de l'être et de l'existant". Une des premières définitions est celle d'Aristote. En effet, Aristote a défini l'Ontologie comme la science de l'Être.On retrouve cette définition dans le Petit Robert [51] avec « Ontologie » : la partie de la métaphysique qui s'intéresse à l'Etre en tant qu'Etre ». « *Mais l'Ontologie est habituellement davantage comprise comme une science des étants que comme une science de l'Etre en tant qu'Etre, c'est-à-dire qu'elle s'intéresse davantage à ce qui existe (les étants ou existants) qu'aux principes de ce qui existe (l'Etre)* » [52]. Cette étude qui explique la réalité par des concepts, des relations et des règles, a depuis lors été reconnue comme ontologie.

L'Ontologie, produit des ensembles : les ontologies. Pour être complet, notons que le mot lui-même date du 17e siècle, avec des dates précises discordantes selon les sources.

Il existe également des définitions de l'ontologie basées sur un domaine spécifique et dépendantes du point de vue de l'utilisateur. Dans [53] l'ontologie est définie comme "*la méthode pour extraire un catalogue des choses ou des entités (C) qui existent dans un domaine (D) selon la perception d'une personne qui se sert d'un certain langage (L) pour le décrire*". Contrairement à la perception "réaliste", la perception du domaine spécifique sur l'ontologie est le principe fondamental pour son application en science de l'information et en ingénierie de connaissance (IC). Une des définitions les plus récentes en IC est que "*l'ontologie définit les termes et les relations de base compris dans le vocabulaire d'un domaine donné aussi bien que les règles pour combiner les termes et les relations pour définir des extensions du vocabulaire*" [54]. Dans [55] Gruber définit l'ontologie comme "*une spécification explicite d'une conceptualisation*". Cette définition est devenue la plus référencée en littérature et par la communauté d'ontologie. Basées sur cette définition, plusieurs autres définitions ont été proposées.

Guarino et Giaretta [56] ont fourni une autre clarification terminologique entre les différentes utilisations de l'ontologie. Ils utilisent les termes : "système conceptuel informel", "état sémantique formel", "spécification d'une conceptualisation", "représentation d'un système conceptuel par l'intermédiaire d'une théorie logique", "vocabulaire utilisé par une théorie logique", et "spécification d'une théorie logique".

Dans [57] Borst et al. ont modifié la définition de Gruber en affirmant que "*les ontologies sont définies comme une spécification formelle d'une conceptualisation partagée*". Dans [58], Swartout et al déclarent que "*une ontologie est un ensemble de termes hiérarchiquement structuré pour décrire un domaine qui peut être utilisé comme base squelettique pour une base de connaissances*".

La notion d'ontologie est héritée d'une tradition philosophique. Elle est ensuite apparue dans le domaine informatique dans les années 1990. Nous pouvons retenir qu'une ontologie est un vocabulaire commun qui définit le sens des concepts et les relations entre ces concepts. Ce vocabulaire peut être associé à un modèle qui décrit le contenu d'une base de connaissances, ses propriétés, la manière dont elle peut être utilisée ainsi que la syntaxe et les contraintes fournies par le langage de représentation. L'objectif est d'assurer la spécification explicite des connaissances au niveau conceptuel en utilisant un langage

formel offrant une sémantique qui peut être plus ou moins rigoureuse, permettant une utilisation non ambigüe des connaissances du domaine [44].

Une ontologie exprime donc une compréhension partagée d'un domaine donné entre un certain nombre d'individus. Un tel accord facilite les communications et les échanges et les rend plus précis et plus efficaces. Ce qui mène, à terme, à d'autres avantages tels que l'interopérabilité, la réutilisation et le partage [50].

Nous avons ici essayé de rassembler les définitions qui nous ont semblé les plus importantes du mot ontologie. D'autres définitions peuvent être trouvées dans la littérature. Ces différentes définitions fournissent des points de vue différents et complémentaires de la même réalité. Certains auteurs fournissent les définitions qui sont indépendantes des processus suivis pour la construction de l'ontologie et de son utilisation dans les applications. Alors que d'autres définitions sont influencées par son processus de développement. Nous pouvons dire que les ontologies visent à capturer la connaissance consensuelle d'une manière générique. Le but est que les représentations puissent être réutilisées et partagées à travers des applications et services par des personnes voire des automates intelligents. Elles sont habituellement construites d'une manière coopérative.

Les méthodologies de construction d'une ontologie peuvent donc influencer les ontologies elles-mêmes. Avant de discuter cet aspect, nous allons classer dans la section suivante les types d'ontologies.

III.2. Types d'ontologies

Plusieurs facteurs peuvent être utilisés pour catégoriser et classer les ontologies Tableau 1 : Classification multicritères des ontologies). Par exemple, selon le type de formalisation, Uschold et Gruninger [59] ont distingué quatre types d'ontologies, en tenant compte du type de langage utilisé pour les implémenter à savoir les ontologies fortement informelles, semi-informelles, semi-formelles et rigoureusement formelles. D'un autre côté, l'équipe de Mizoguchi [60] et de Van Heijst [61] fournissent une typologie exhaustive des ontologies basée sur la nature de la conceptualisation. Ainsi, ils distinguent les ontologies de représentation de connaissance, les ontologies génériques, les ontologies de haut niveau, les ontologies de domaine, les ontologies de tâches, de méthodes et linguistiques. Dans [61],

une autre catégorisation des ontologies est faite selon leur propos. Lassila et McGuiness [62], ont classifié les différents types d'ontologies selon le niveau de complexité, allant des plus légères ou "LightWeight ontologies" (exemples : catalogues, glossaires, thésaurus, etc.) aux plus complexes ou "HeavyWeight ontologies".

La catégorisation selon le nombre de points de vue de concepteurs donne deux types d'ontologies [63]. Les travaux de Keita ont amené une autre classification selon l'état (consensuel ou pas) des définitions.

Toutes ces classifications sont résumées dans le tableau suivant.

Classification selon	Types d'ontologies
Formalisation	- **Des ontologies fortement informelles** (si elles sont écrites dans un langage naturel) - **Des ontologies semi-informelles** (si elles sont exprimées dans une forme restreinte et structurée de langage naturel ; en utilisant des modèles) - **Des ontologies semi-formelles** (si elles sont définies dans un langage artificiel et formellement défini ; exemple : Ontolingua, OWL) - **Des ontologies rigoureusement formelles** (si elles sont définies dans un langage avec la sémantique formelle, les théories et les preuves des propriétés telles que la solidité et la perfection.).
Type de conceptualisation	- **Les ontologies de représentation de connaissance** (selon [61] ces ontologies définissent des primitives de représentation utilisées pour formaliser la connaissance avec un paradigme donné ; exemples : « Frame Ontology » [55] et la « OKBC[8] Ontology »). - **Les ontologies génériques** (sont utilisées pour représenter la connaissance commune, ou bien consensuelle, réutilisable dans les domaines. Ces

[8] Ontology Knowledge Base Connectivity

	ontologies incluent le vocabulaire lié aux choses, aux événements, au temps, à l'espace, à la causalité, au comportement, à la fonction, à la méréologie, etc. Exemples : la « Mereology Ontology » [57] et la « gène-ontologie »[9]). - **Les ontologies de haut niveau** ("Top-level ou Upper-level ontologies" ; décrivent des concepts très généraux et fournissent des notions générales sous lesquelles tous les termes racines dans les ontologies existantes devraient être liés. Exemples : SOWA[10], CYC[11] et SUO[12]). - **Les ontologies de domaine** (qui sont réutilisables dans un domaine spécifique donné. Elles fournissent les vocabulaires sur les concepts et leurs relations dans un domaine, sur les activités qui ont lieu dans ce domaine, et sur les théories et les principes élémentaires régissant ce domaine). [60] - **Les ontologies de tâche** (décrivent le vocabulaire relié à une tâche générique ou à une activité en spécialisant les termes dans les ontologies de haut niveau. Elles fournissent un vocabulaire systématique des termes utilisés pour résoudre les problèmes liés aux tâches qui peuvent ou ne peuvent pas appartenir au même domaine.) [60] - **Les ontologies de méthodes** (donnent les définitions des concepts et des relations appropriés appliquées pour spécifier un processus de raisonnement afin de réaliser une tâche particulière). - **Les ontologies linguistiques** (Le but de ce type d'ontologie est de décrire les constructions sémantiques plutôt que de modéliser un domaine. La principale caractéristique de ces ontologies est qu'elles sont liées à la sémantique des unités grammaticales (mots, groupes nominaux, adjectifs, etc.). Exemples : EuroWordnet, WordNet.)
Propos	- **Les ontologies d'application** (sont dépendantes des applications. Elles contiennent toutes les définitions nécessaires pour modéliser la

[9] An Introduction to Gene Ontology, [en ligne le 11/07/2012] : http://www.geneontology.org/
[10] [en ligne le 12/07/2012] : http://www.jfsowa.com/ontology/
[11] Dérivé de « encyclopédie », et prononcé *saïk* est une marque déposée par Cycorp, Inc.
[12] Standard Upper Ontology

	connaissance requise pour une application particulière. Les ontologies d'application étendent et spécialisent souvent le vocabulaire des ontologies de domaine et de tâche pour une application donnée.) - **Les ontologies de référence** (qui sont proposées dans la perception philosophique comme des théories pour "des réalités indépendamment existantes"). Par opposition à formelle, l'ontologie axiomatisée constitue une description détaillée du domaine. Les ontologies de référence permettent de clarifier les définitions entre les communautés en expliquant la signification des termes qui sont inclus dans l'ontologie de haut niveau "top-level" [50].
Niveau de complexité	- **« LightWeight ontologies »** (Les plus légères, exemples : catalogues, glossaires, thésaurus, etc.). - **« HeavyWeight ontologies »** (Les plus lourdes, exemples : ontologies expressives avec des axiomes et fonctions.).
Nombre de points de vue des concepteurs	- **Les ontologies inspirationnelles** (dans le cas où l'ontologie est conçue selon le point de vue individuel d'un concepteur sur le domaine) - **Les ontologies collaboratives** (c'est le cas de la conception avec de multiples points de vue de différents concepteurs et acteurs sur le domaine) [63]
Etat	- **Les ontologies post- consensuelles** (lorsque le consensus est déjà atteint sur les définitions. Ainsi, dans la conception d'ontologie post-consensuelle, l'ontologie est écrite dans un langage tel que les logiques de description dans lesquelles certaines caractéristiques déductives sont importantes pour assurer l'interopérabilité, particulièrement en servant de base à la génération de médiateur.) - **Les ontologies pré-consensuelles** (lorsque le consensus n'est pas encore atteint sur les définitions. Ce type d'ontologies se caractérise par son appartenance aux premières phases de construction des ontologies. Dans ces phases, les définitions sont principalement informelles et multiples et constituent seulement un réseau sémantique).

Tableau 1 : Classification multicritères des ontologies

III.3. Composants d'une Ontologie

Une ontologie peut être vue comme un treillis de concepts et de relations [64]. Ces concepts sont destinés à représenter les objets du monde sous une forme compréhensible aussi bien par les hommes que par les machines. Si certaines divergences relatives à la structure (degré de formalisation) de l'ontologie peuvent être constatées, les composants d'une ontologie sont toujours les mêmes [65] Les principaux composants d'une ontologie sont les suivants :

- Les concepts : ce sont des notions (ou objets) permettant la description d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement. Les concepts peuvent être abstraits ou concrets, élémentaires ou composés, réels ou fictifs. Habituellement, les concepts sont organisés en taxonomie. C'est une hiérarchie de concepts (ou d'objets) reliés entre eux en fonction de critères sémantiques particuliers.
- Les relations : ce sont les liens organisant les concepts d'un domaine. Elles sont formellement définies comme tout sous-ensemble d'un produit de n ensembles. Des exemples de relations binaires sont : « sous-concept-de », « connecté-à », « sorte-de », « est-un », etc.
- Les propriétés (ou attributs) se sont des restrictions des concepts ou des relations.
- Les fonctions : ce sont des cas particuliers des relations dans lesquelles le $n^{ième}$ élément de la relation est unique pour les n-1 précédents. Un exemple de fonction binaire est la relation « mère-de ».
- Les axiomes de l'ontologie : ils permettent de définir la sémantique des termes (classe, relations), leurs propriétés et toutes contraintes quant à leur interprétation. Ils sont définis à l'aide de formules bien formées de la logique du premier ordre en utilisant les prédicats de l'ontologie.

III.4. Différence entre Ontologies et Bases de Connaissances

Les notions d'Ontologie et de Base de Connaissances (KB : Knowledge Bases) peuvent prêter à confusion, surtout qu'il se basent sur les mêmes [66].

On retrouve en effet les notions utilisées : mêmes langages (OWL, RDF-S, etc.), mêmes outils et infrastructures. Mais tout fichier OWL, par exemple, n'est une ontologie.

En général les ontologies sont composées d'un vocabulaire et de la spécification formelle de ce dernier alors que les bases de Connaissance sont composées d'instances.

Pour mieux comprendre cette différence, prenons l'exemple suivant : une ontologie définit des concepts tels que « Homme » et « Femme » et explicite que ces deux concepts s'excluent mutuellement. Les individus « Peter », « Bob » et « Marie » peuvent être des instances des concepts « Homme » et « Femme ». Ils ne constituent pas une partie de l'ontologie. Mais peuvent peupler une base des connaissances ou une base de données.

Cette distinction entre les notions d'ontologie et de bases des connaissances n'est pas toujours évidente. Il est possible que l'on retrouve dans une ontologie, des instances (individus) de concepts. En effet, dans certains cas, des instances peuvent faire partie de la spécification formelle du domaine. Ceci dépend essentiellement des objectifs et de la portée de l'ontologie. Des instances peuvent servir à mieux décrire ou définir certains concepts du domaine concerné. A titre d'exemple, la ville « Paris » qui est une instance du concept (classe) « ville » doit appartenir par exemple à une ontologie du « tourisme », mais un train spécifique qui emmène à « Paris » ou bien un métro qui circule dans cette ville n'appartiennent pas à cette ontologie.

IV. Les langages utilisés pour la description sémantique

Le langage de spécification est l'élément central sur lequel repose l'ontologie. La plupart de ces langages se basent sur la logique du premier ordre, et représentent donc les connaissances sous forme d'assertion (sujet, prédicat, objet).

Dans la littérature, plusieurs langages ont été proposés pour décrire des ontologies. Certains ont été définis par des Industriels ou des regroupements d'industriels, d'autres par des organismes de normalisation.

Les plus utilisés de ces langages sont les suivants :

- OWL (Ontology Web Language) [13]: c'est un langage proposé initialement pour faire des rapports ontologiques. Il a été développé par extension des langages RDF (Resource Description Framework) et de RDFS (RDF Schema), et en s'inspirant des projets OIL (An Ontology Infrastructure for the Semantic Web), DAML (DARPA Agent Markup Language) et DAML+OIL. OWL a été très utilisé avec le World Wide Web pour définir ses différents éléments (classes, propriétés et individus) comme ressources RDF et en les identifiant par des URIs (Uniform Resource Identifier).
- KIF (Knowledge Interchange Format)[14] est une syntaxe permettant de décrire la logique de premier ordre et basée sur des S-expressions.
- Le projet Cyc[15] a proposé son propre langage d'ontologie appelé CycL (Cyc Language). Ce dernier est également basé sur le calcul du premier ordre.
- Rule Interchange Format (RIF)[16] et F-Logic[17] combinent ontologies et règles.
- Le langage Gellish (A Generic Extensible Ontological Language) est un langage open source pour les réseaux et le stockage des données pour les systèmes autonomes. Il inclut des règles pour sa propre prolongation et intègre ainsi une ontologie avec une langue d'ontologie [67].

Parce que plus expressif que son prédécesseur RDFS, le langage OWL est désormais le standard le plus utilisé dans le paysage des ontologies. Grâce à sa sémantique formelle basée sur une fondation logique largement étudiée, OWL permet de définir des associations plus complexes des ressources ainsi que les propriétés de leurs classes respectives.

OWL définit trois sous-langages : OWL-Lite, OWL-DL et OWL-Full, du moins expressif au plus expressif.

[13] Mindswap : first site on the Semantic Web. En ligne 08/12 http://www.mindswap.org.
[14] KIF : Knowledge Interchange Format. En ligne 08/12 http://logic.stanford.edu/kif/dpans.html
[15] CycL : en ligne 08/12 http://www.cyc.com/cycdoc/ref/cycl-syntax.html
[16] RIF : http://www.w3.org/2005/rules/wiki/RIF_Working_Group
[17] F-Logic : http://flora.sourceforge.net/

V. La logique de description

Les logiques de description (DL : Description Logic) sont une famille de langages permettant de représenter, d'une manière formelle et structurée, la connaissance terminologique d'un domaine d'application. Elles se rapportent, d'une part à la description des concepts utilisés pour décrire un domaine, et d'autre part à la sémantique basée sur la logique qui peut être donnée par une transcription en logique des prédicats du premier ordre[18].

La plupart des logiques de description classent la connaissance en deux groupes :

- Les informations terminologiques: ce sont les définitions des notions de base ou les définitions dérivées et de la façon par laquelle ces informations sont reliées entre elles. Les informations peuvent être "génériques" ou "globales", mais vraies dans tous les modèles et pour tous les individus.
- les informations sur les assertions: ces informations sont "spécifiques" ou "locales". Elles peuvent être vraies pour certains individus uniquement.

Les informations connues sont modélisées par un couple (T, A), où T est un ensemble de formules relatives aux informations terminologiques (désignée par T-Box) et A est un ensemble de formules relatives aux informations sur les assertions (désignée par A-Box). [15]

Les logiques de description utilisent les notions de concept, de rôle et d'individu.

- Les concepts correspondent à des "classes d'éléments" et sont interprétés comme un ensemble dans un univers donné.
- Les rôles correspondent aux "liens entre les éléments" et sont interprétés comme des relations binaires sur un univers donné.
- Les individus correspondent aux éléments d'un univers donné.

La logique de description est usuellement la fondation théorique pour les ontologies. Les sous-langages OWL-DL et OWL-Lite sont basés sur la logique de description.

[18] Deborah L. McGuinness, Daniele Nardi, Peter F. Patel-Schneider. THE DESCRIPTION LOGIC HANDBOOK: Theory, implementation, and applications. s.l.: Franz Baader, 2002.

VI. Des règles pour une meilleure inférence :

Des règles peuvent être appliquées pour réaliser le système ou le service dynamique (qui emploient l'ontologie). Le service ou le processus pourrait se comporter différemment en fonction des circonstances variables définies par les valeurs et liens contextuels des éléments de l'ontologie considérée.

Les règles prennent généralement la forme d'une implication entre l'antécédent (corps) et le conséquent (entête). Elles peuvent être interprétées comme : toutefois que les conditions indiquées dans la prise antécédente se sont tenues, alors les conditions indiquées dans le conséquent doivent également se tenir.

SWRL[19] (A Semantic Web Rule Language) est un bon exemple des langages de description de règles. C'est une proposition combinant les sous-langages d'ontologie de Web, OWL (OWL DL et OWL Lite), avec ceux de RuleML (Rule Markup Language).

La Figure 2 **:** Exemple de règle SWRLFigure 2 illustre un exemple simple de règle (SWRL).

```
hasParent(?x1,?x2) ∘ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)
```

Figure 2 **:** Exemple de règle SWRL

Sur l'exemple de la Figure 2 **:** Exemple de règle SWRL, la règle stipule que si x1 a pour parent x2, et si x3 est frère de x2, alors x1 a pour oncle x3.

Une des propositions pratiques[20], les langages d'ontologie tels qu'OWL, ainsi que les règles associées (exprimées par exemple en SWRL) sont sérialisés (i.e. encapsulés) en XML (Extensible Markup Language). De même la description logique est encapsulée dans OWL enrichi avec les règles exprimées en SWRL (c.f. Figure 3).

[19] SWRL : W3C Member Submission, [En ligne] 05 2008. http://www.w3.org/Submission/SWRL/.

[20] The Semantic Web: From Teaching To Research Yuh-Jong H, Department of Computer Science

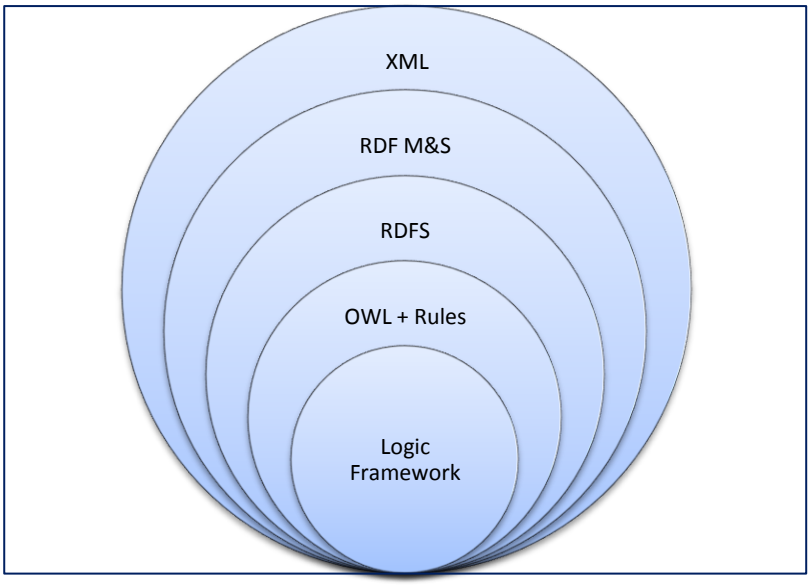


Figure 3 : Encapsulation des langages d'ontologies dans l'XML

VII. Méthodologies de conception et de développement d'une ontologie

L'élaboration de toute ontologie doit s'appuyer sur un certain nombre de règles à respecter représentant la méthodologie de construction.

Concernant les ontologies, il n'existe pas encore de méthodologie bien rodée. Cependant, l'analyse de la littérature et des différents travaux réalisés jusqu'ici peut nous donner des renseignements sur les principaux aspects à considérer et quelques règles à respecter [44].

Nous énumérons ici quelques critères et règles que nous recommandons et que nous avons suivis pour développer l'ontologie que nous allons proposer dans la section suivante :

- La clarté et l'objectivité : l'ontologie doit fournir le sens des termes qu'elle utilise en donnant des définitions des plus objectives possibles permettant même de leur associer une documentation en langage naturel.
- L'exhaustivité : une définition exprimée par une condition nécessaire et suffisante est préférable à une définition exprimée seulement par une condition nécessaire ou par une condition suffisante.

- La cohérence : bien vérifier la cohérence des règles afin de ne pas induire inférences (raisonnements) non cohérentes avec les définitions.
- L'extensibilité monotone maximale : l'ontologie doit permettre l'ajout de nouveaux termes sans la nécessité de modifications dans les définitions existantes.

Partie 2 : Ubiquity-Ont : une ontologie pour les systèmes Ubiquitaires

I. Introduction

Dans la littérature, il existe plusieurs efforts de développement d'ontologies pour des sous domaines des TIC (réseaux [68], terminaux [69], mobilité, sécurité géo-localisation [70], radio, services, bases des données [], …). Malheureusement, ces ontologies restent spécifiques et peu génériques. A titre d'exemple, nous pouvons citer l'ontologie nommée « Mobile Ontology » [71] proposée dans le cadre du projet européen FP6-IST-SPICE [21]. Cette ontologie est proposée essentiellement pour standardiser le format des échanges entre les composants de la plateforme de ce projet et pour permettre le raisonnement. Elle classifie les entités de l'environnement en deux catégories : des entités réelles (spacial entities) telles que les équipements mobile, les personnes, les localisations (places) ; et et des entités virtuelles (virtual entities) telles que les services, les utilisateurs et les groupes d'utilisateurs. Malgré qu'elle soit restreinte au domaine des télécommunications mobiles, cette ontologie est conçue pour être extensible et coopérer avec d'autres sous-ontologies (sub-ontologies) comme celle pour décrire les contenus (content sub-ontology) ou bien celle pour décrire les services (service sub-ontology).

Dans cette partie, nous proposons une ontologie pour le domaine des TIC centrée sur l'utilisateur. Cette ontologie a pour objectif de définir un environnement générique pour les réseaux et services permettant de faciliter les opérations de configuration, de reconfiguration, de création et d'adaptation de services dans les systèmes ubiquitaires. Nous donnons le nom *Ubiquity-Ont* à cette ontologie qui sera générique et utile pour les systèmes ubiquitaires dans les environnements TIC hétérogènes.

[21] http://www.ist-spice.org

Nous avons adopté pour cela une approche itérative car nous considérons que l'ontologie suit un processus d'évolution, permettant de l'affiner et de l'enrichir par de nouveaux concepts et de nouveaux détails.

Les différentes étapes de ce processus de création de *Ubiquity-Ont* seront explicitées dans ce qui suit.

Mais avant d'aborder ce processus, voici un certain nombre de règles fondamentales que nous nous sommes imposées pour cette conception et auxquelles nous nous référerons plusieurs fois. Ces règles peuvent paraître dogmatiques, mais comme nous allons le voir plus loin, elles nous seront utiles à plusieurs occasions pour la prise de certaines décisions conceptuelles.

1. Il n'existe pas une seule façon « correcte » pour modéliser un domaine - il peut toujours y avoir des alternatives aussi viables et défendables. La « meilleure solution », si elle existe, dépend presque toujours de l'application ou du service visé et de ses évolutions et que nous pouvons essayer d'anticiper ou de prévoir. Une vision large et de long terme est donc utile à la conception d'une « bonne » ontologie.
2. Le développement d'une ontologie est un processus itératif, qui nécessite plusieurs passages afin de l'affiner et de le corriger.
3. Les concepts dans une ontologie, objets et relations, doivent être très proches des objets réels (physiques ou logiques) et de leurs relations dans le domaine concerné, privilégier des noms (objets) ou verbes (relations) naturels et des phrases qui décrivent ce domaine.

L'explicitation du but de l'ontologie et de son degré de finesse (détaillée ou générale), guidera la plupart des décisions de modélisation tout au long du processus. Parmi les alternatives les plus viables, nous aurons besoin de déterminer la plus adaptée à la tâche que nous nous sommes fixée, la plus intuitive, la plus extensible et la plus réalisable en termes de maintenance. Il faut également se rappeler qu'une ontologie est un modèle de la réalité du monde et que les concepts dans l'ontologie doivent refléter cette réalité. Après avoir défini une version initiale de l'ontologie, nous pouvons l'évaluer et la mettre au point en l'utilisant dans un système qui résout un certain nombre de problématiques du domaine concerné. A plusieurs reprises, ne pas hésiter à réviser l'ontologie initiale pour l'enrichir et

l'améliorer. Ce processus de conception itérative continuera tout au long du cycle de vie de l'ontologie.

II. Méthodologie adoptée pour concevoir notre ontologie

Dans cette section, nous allons concevoir l'ontologie *Ubiquity-Ont* que nous proposons pour le domaine des TIC tout en explicitant la méthodologie suivie. Cette méthodologie sera générique et utilisable par les chercheurs de la communauté des concepteurs des ontologies de n'importe quel domaine.

Notre objectif c'est de décrire le contexte d'une façon sémantique compréhensible par la machine afin de développer des applications qui utiliseront ces informations pour s'adapter au contexte. Notre ontologie doit alors bien décrire le domaine des réseaux, des équipements utilisateurs qui se connectent à ces derniers, des services offerts à travers ces réseaux, ainsi que les préférences, les exigences et les capacités de ces concepts. Cette ontologie doit répondre aux requêtes de ces applications. *Ubiquity-Ont* semble bien être une base de données contextuelle, mais le fait que ces données sont décrites sémantiquement rendra de cette ontologie une base de connaissances plus que de simples données.

Dans ce qui suit, nous utiliserons la terminologie suivante :

- Classe : Concept, Objet, ensembles, collections, ou types d'objets.
- Attribut : propriétés, fonctionnalités, caractéristiques ou paramètres que les objets peuvent posséder et partager ;
- Relation : les liens que les objets peuvent avoir entre eux ;
- Instance : individus, les objets de base.

II.1. Critères d'évaluation d'une ontologie

D'après Gruber [55], pour évaluer et mettre en évidence les aspects importants d'une ontologie il suffit d'évaluer cinq critères à savoir :

- la cohérence : aucune contradiction entre les définitions des concepts ;
- la clarté : indépendamment du contexte, un concept défini dans l'ontologie doit faire passer le vrai sens du terme avec une définition complète et objective ;
- l'extensibilité : on doit pouvoir ajouter de nouveaux concepts sans avoir à toucher aux structures de l'ontologie ;
- une déformation d'encodage minimale : il faut minimiser les définitions simplistes et qui déforment le sens réel du terme. C'est-à-dire il faut augmenter toujours le degré de conceptualisation ;
- un engagement ontologique minimal : le but d'une ontologie est de définir un vocabulaire pour décrire un domaine, si possible de manière complète ; ni plus, ni moins. Contrairement aux bases de connaissances par exemple, on n'attend pas d'une ontologie qu'elle soit en mesure de fournir systématiquement une réponse à une question arbitraire sur le domaine. Toujours selon Gruber, « l'engagement ontologique peut être minimisé en spécifiant la théorie la plus faible (celle permettant le plus de modèles) couvrant un domaine ; elle ne définit que les termes nécessaires pour partager les connaissances consistantes avec cette théorie ».

II.2. Première étape : identifier le domaine

Dans cette première étape, il est important de commencer par définir le domaine visé par l'ontologie avec toute sa portée incluant son utilisation et ses utilisateurs potentiels. Pour ce faire, il faudra répondre aux quelques questions de base suivantes :

- Quel est le domaine que va couvrir l'ontologie ?
- Dans quel but utiliserons-nous l'ontologie ?
- A quels types de questions l'ontologie devra-t-elle fournir des réponses ?
- Qui va utiliser et maintenir l'ontologie ?

Avant d'atteindre un stade de maturité de l'ontologie, les réponses à ces questions peuvent varier au cours du processus de conception. Les réponses peuvent parfois aider à étendre ou à limiter la portée initialement prévue.

L'évolution des réponses à ces questions induit un processus itératif permettant d'affiner l'ontologie qui concerne dans notre cas un environnement hétérogène de réseaux de télécommunications.

Dans cet environnement, plusieurs concepts peuvent être envisagés, allant des terminaux utilisateurs aux services offerts. Certains feront partie de *Ubiquity-Ont* et d'autres seront écartés. Nous expliciterons le pourquoi de ces choix par la suite.

- Les utilisateurs,
- les terminaux clients et leurs caractéristiques,
- les réseaux de télécommunication, leurs technologies et leurs équipements,
- les services disponibles, leurs caractéristiques y comprises leurs capacités d'adaptation ainsi que leurs exigences en termes de qualité de service,
- la notion de compatibilité des terminaux utilisateurs avec les technologies réseaux
- la notion de QoS,
- les préférences utilisateur, leurs exigences en termes de QOS, de coût, de consommation d'énergie.

En même temps, notons qu'il est peu probable que la conception d'une ontologie pour un besoin donné, puisse inclure tous les concepts du domaine. L'objectif, n'est pas ici de définir une ontologie universelle mais réutilisable et extensible (voir étape 2). En effet, une ontologie bien définie pourra être plus facilement étendue à d'autres besoins du domaine et même liée via des classes et des relations à d'autres ontologies complémentaires.

Ainsi, nous nous limitons volontairement aux aspects des besoins visés. Dans le cas de cette thèse, on se limitera donc aux besoins définis par le domaine visé qui est les réseaux et services de télécommunications, et en particulier sur l'offre de services, la mobilité et la qualité de service. On essayera dans un second temps d'étendre *Ubiquity-Ont* aux aspects de gestion de la consommation d'énergie sur les terminaux.

Les aspects liés à la consommation d'énergie seront donc ignorés dans un premier temps car on supposera qu'ils ne font pas partie des besoins exprimés. D'autres aspects sont également ignorés tels que ceux liés à la gestion et à la maintenance des équipements réseaux, à la gestion des ventes des services et à la facturation.

Une des techniques permettant de déterminer la portée d'une ontologie est de rédiger une liste de questions auxquelles une base de connaissances fondée sur une ontologie devrait pouvoir répondre. Cette liste est appelée questions de compétence par Gruninger et Fox [72].

Ces mêmes questions serviront plus tard pour effectuer des tests de vérification à posteriori pour savoir si l'ontologie contient suffisamment d'informations pour répondre aux besoins exprimés. La liste de questions et leurs réponses permettent aussi de déterminer le niveau de détail, ou le besoin de représenter ou non des domaines particuliers liés au domaine principal. La liste doit être la plus large possible mais pas nécessairement exhaustive. Le processus itératif va ensuite aider à l'améliorer.

Dans le domaine considéré dans cette thèse, un exemple de questions de compétences possible est :

- « Quelles sont les informations sur lesquelles un algorithme de handover vertical doit se baser ? »,
- « Comment choisir la classe de service la plus adaptée à un contexte donné ? »,
- « Comment garantir la continuité de service tout au long de la session utilisateur en cas de handover vertical? ».

II.3. Deuxième étape : Envisager l'éventuelle réutilisation des ontologies

Il est toujours utile de prendre en considération les ontologies existantes et d'examiner s'il est possible de les réutiliser même à les étendre et les compléter en fonction des nouveaux besoins.

La réutilisation des ontologies existantes peut aussi prendre une autre forme qui est l'interaction entre les ontologies via des relations.

Exemple d'interaction avec des ontologies existantes : la base WURFL

Dans le domaine que nous considérons, nous avons envisagé de modéliser dans l'ontologie les terminaux mobiles et leurs caractéristiques. Or il existe déjà une base appelée WURFL [22] qui consiste en un fichier XML, mis à jour régulièrement par une communauté ouverte mais spécialisée. La base WURFL contient des informations décrivant les capacités des équipements et les caractéristiques de plusieurs mobiles classés par marques et modèles.

[22] Wireless Universal Resource FiLe

Cette base est publiée sous une licence de logiciels libres dès sa version 2.2. En 2011, les contributeurs de ce projet communautaire ont publié une API web faite disponible avec le langage PHP. Cette API (Application Programming Interface) est disponible avec la licence AGPL (GNU[23] Affero General Public License) permettant toute utilisation non commerciale.

Pour chaque équipement mobile, WURFL offre plus que 500 caractéristiques. Les mobiles sont classés en 30 groupes. En mars 2012, un nouveau service payant a vu le jour appelé « WURFL Cloud »[24] permettant de détecter l'équipement connecté et ses caractéristiques et offrant la possibilité d'adapter les services en fonction.

D'autres bibliothèques d'ontologies réutilisables existent et sont mises à la disposition de la communauté. La bibliothèque d'ontologies Ontolingua[25] par exemple offre des outils facilitant l'exploration, la création, l'édition et l'utilisation des ontologies. Aussi, la bibliothèque d'ontologies DAML [26] est très riche et modélise plusieurs concepts du web sémantique.

Exemple d'extension d'ontologie existante

Une ontologie existante peut être étendue pour s'enrichir de nouveaux concepts ou relations non considérés initialement. Un exemple, qui sera détaillé dans le chapitre cinq est l'extension de l'ontologie que nous proposons, *Ubiquity-Ont*, pour les réseaux et services par des concepts liés à la consommation d'énergie.

II.4. Troisième étape : Enumérer les concepts de base de l'ontologie

Après avoir passé les deux premières étapes, il est utile maintenant de noter, sous forme de liste, tous les termes à traiter par l'ontologie à définir, *Ubiquity-Ont*.

[23] GNU : un système d'exploitation libre lancé en 1984
[24] http://www.scientiamobile.com/cloud
[25] http://www.ksl.stanford.edu/software/ontolingua/
[26] http://www.daml.org/ontologies/

Cette liste permet de recenser les termes auxquels on s'intéresse, leurs propriétés et relations.

Dans le cadre du domaine qu'on vise dans cette thèse, les termes qui paraissent les plus importants incluent : réseau, utilisateur, terminal mobile, mobilité, couverture, handover, localisation géographique, équipement, capacité, limitations, débit, QoS, horaire actuel, etc.

L'analyse de ces termes va ensuite ressortir les concepts des propriétés et relations. Ainsi, nous pouvons dans un premier temps considérer que quatre concepts ressortent, les autres termes indiquant des propriétés ou des relations (c.f. Figure 4).

- Utilisateur (user)
- Terminal utilisateur (device)
- Réseau (network)
- Service

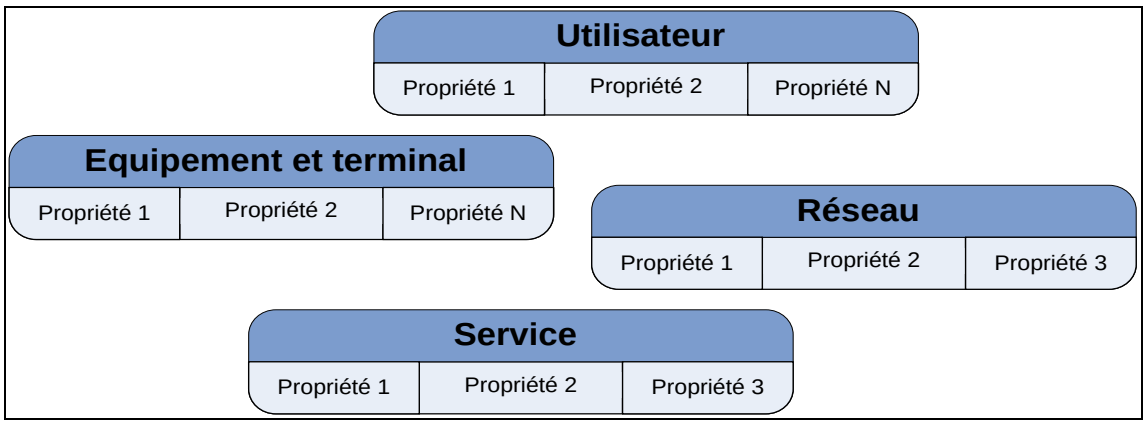


Figure 4 : Premiers concepts importants dans *Ubiquity-Ont*

La liste établie dans cette étape ainsi que les principaux concepts identifiés, constituent le socle de l'ontologie. Dans la prochaine étape, nous allons pouvoir affiner *Ubiquity-Ont*.

II.5. Quatrième étape : affiner l'ontologie et ses concepts

Cette phase va nous permettre, à partir des concepts de base identifiés dans l'étape précédente, d'affiner et de compléter l'ontologie visée

Dans la littérature, plusieurs approches sont proposées pour développer une hiérarchie de classes. On se basera principalement sur celle proposée par Uschold et Gruninger 1996 [73] qui définit les procédés suivants :

- Un procédé de développement de haut en bas commence par une définition des concepts les plus généraux du domaine et se poursuit par la spécialisation des concepts. Par exemple, nous pouvons commencer par créer des classes pour les concepts généraux ou importants que nous avons énumérés dans l'étape précédente Utilisateur, Equipement utilisateur, Réseau, Service. Puis nous spécialisons la classe Utilisateur en créant quelques-unes de ses sous-classes : Utilisateur Grand public, Utilisateur professionnel, Utilisateur abonné, Utilisateur invité, etc. Nous pouvons en outre catégoriser la classe Utilisateur Abonné, par exemple Abonné Wifi/3G, Abonné Wifi seulement, Abonné 3G seulement, et ainsi de suite. Un autre exemple c'est de définir le concept « protocoles réseaux » pour passer par la suite aux différents protocoles réseaux héritant de la classe parent, tel que les protocoles IP, TCP, UDP, etc. (c.f. Figure 7).
- Un procédé de développement de bas en haut commence par la définition des classes les plus spécifiques, les feuilles d'une hiérarchie, et se poursuit avec le regroupement de ces classes en concepts plus généraux. Par exemple, nous pouvons commencer en définissant des classes pour la Technologie Wifi et celle des Technologie 3G (c.f. Figure 6. Nous pouvons ensuite créer une classe commune Réseaux Sans fil qui à son tour est une sous-classe de la classe Réseau.

- Un procédé combiné de développement est une combinaison des deux approches, de haut en bas et de bas en haut. Au tout début, les concepts les plus saillants sont définis, ensuite ils sont généralisés ou spécialisés, suivant le cas. Nous pourrions commencer par quelques concepts du haut niveau tels que Réseau, Utilisateur et quelques concepts spécifiques, tels que Smartphone 3G. Puis, on peut les mettre en relation avec d'autres concepts de niveau intermédiaire, tel que Smartphone. Ensuite, on peut poursuivre en créant toutes les classes des réseaux et des

utilisateurs ainsi que les types de Smartphones, en créant également de ce fait, tout un ensemble de concepts de niveau intermédiaire qui atteint enfin les concepts de haut niveau tel que « *Equipement Utilisateur » (ou Device* c.f. Figure 5)**Erreur ! Source du renvoi introuvable.**.

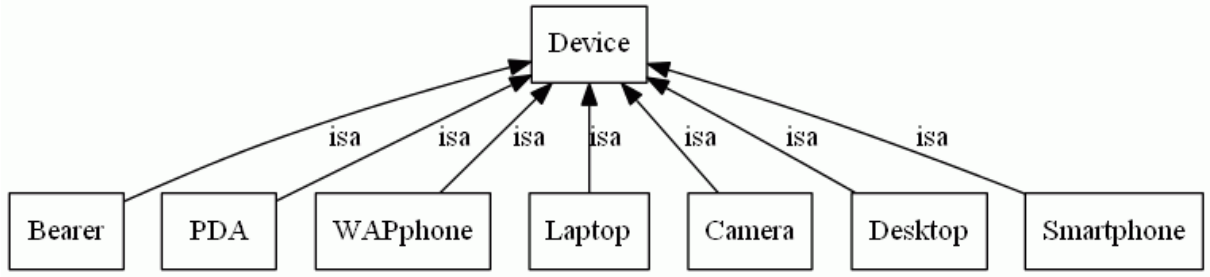


Figure 5 : Développer la hiérarchie de la classe « *device* » de *Ubiquity-Ont*

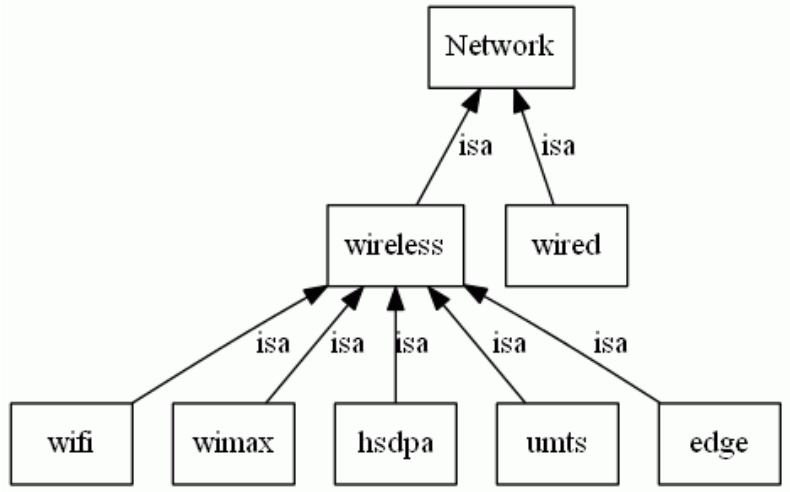


Figure 6 : Développer la hiérarchie de la classe « *network* » de *Ubiquity-Ont*

Aucune de ces trois méthodes n'est fondamentalement meilleure que les autres. L'approche à adopter dépend fortement du point de vue du concepteur sur le domaine. L'approche combinée est souvent, la plus facile à utiliser pour la plupart des développeurs d'ontologies, étant donné que les concepts « du milieu » ont tendance à être les concepts les plus descriptifs du domaine [74]. Nous avons choisi d'adopter cette méthode tout au long de la définition de la hiérarchie et des classes de notre ontologie, surtout que cette thèse a été effectuée dans le cadre de travaux collaboratifs, impliquant plusieurs spécialistes venant de plusieurs domaines.

Dans la liste créée pendant la troisième étape, nous avons sélectionné les termes qui décrivent des objets ayant des existences indépendantes plutôt que les termes qui décrivent ces objets.

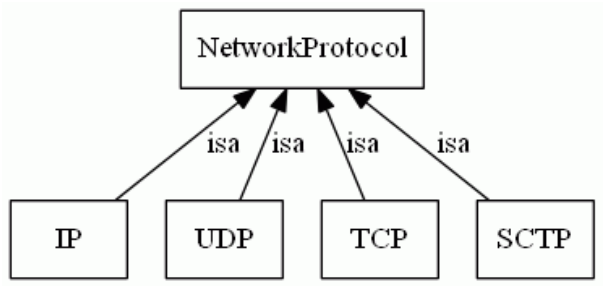


Figure 7 : Développer la hiérarchie de la classe « network protocol » *Ubiquity-Ont*

Ces termes constitueront les classes dans l'ontologie *Ubiquity-Ont* et deviendront des points d'ancrage dans la hiérarchie des classes. Nous organisons ensuite les classes dans une taxonomie hiérarchique en nous demandant, si en étant instance d'une classe, un objet sera nécessairement (c'est à dire par définition) une instance d'une autre classe.

« Si une classe A est super-classe d'une classe B, alors toute instance de B est également, une instance de A. »

En d'autres termes, la classe B représente un concept qui est également une sorte de A.

Par exemple, dans notre ontologie, chaque Point d'Accès WiFi est obligatoirement un Réseau. Par conséquent, la classe Point d'Accès WiFi est une sous-classe de la classe Réseau.

II.6. Cinquième étape : Définir les propriétés des classes – attributs

Les classes seules ne fournissent pas assez d'information pour répondre aux questions de la première étape. Après avoir défini les classes définissant les principaux concepts, nous devons décrire maintenant de façon détaillée la structure interne de ces concepts.

Dans les étapes précédentes, nous avons déjà sélectionné des classes à partir de la liste des termes parlants identifiés du domaine. La plupart des termes restants ont de fortes chances d'être des propriétés de ces classes ou des relations entre elles.

Dans l'exemple considéré, les termes que nous avons évoqué dans l'étape 3 étaient : *réseau, utilisateur, terminal mobile, mobilité, couverture, handover, localisation géographique, équipement, capacité, limitations, débit, QoS, horaire actuel*

Certains de ces termes ont été traduits en concepts. Les autres vont maintenant être traduits en propriétés ou en relations.

En général, il existe plusieurs types de propriétés :

- Propriétés intrinsèques, telle que la taille de l'écran d'un équipement utilisateur, la bande passante supportée par le réseau ou bien plus dynamiquement la liste des réseaux disponibles à un instant donné (cf. Figure 8)
- Propriétés extrinsèques, tel que le nom d'un utilisateur et ces abonnements, l'APN (Acces Point Name) d'un réseau 3G donné, l'identifiant d'une cellule réseau, etc. (c.f. Figure 9).
- Parties de concept, si l'objet est structuré ; elles peuvent être des « parties » physiques ou abstraites (ex : les composants et sous composants d'un service ou d'un équipement donné, c.f. Figure 10Figure 10 : Exemples d'attributs de la classe équipement (device) de *Ubiquity-Ont*).
- Relations entre concepts et instances de concepts: ce sont les relations entre les membres individuels d'une classe et les autres entités (ex : le constructeur d'un terminal mobile représentant une relation entre le modèle de ce dernier et la marque constructeur). Ces relation seront introduites et enrichies dans l'étape suivante.

Lors de cette étape, il est possible de définir de nouveaux concepts car on découvre que certains termes ne peuvent être ni attribut ni relation de concepts existants.

Il est également possible que certains concepts aient des propriétés et des informations de contexte communes du point de vue conceptuel. Nous pouvons dans ces cas définir de nouveaux concepts de haut niveau puis hériter de leurs propriétés. A titre d'exemple, on peut définir le concept de haut niveau « ressource » qui sera hérité par la suite par des concepts de ressources spécifiques aux terminaux ou aux équipements réseaux.

Lors ce cette étape, il est possible de définir de nouveaux concepts car on découvre que certains termes ne peuvent être ni attribut ni relation de concepts existants.

Il est également possible que certains concepts aient des propriétés et des informations de contexte communes du point de vue conceptuel. Nous pouvons dans ces cas définir de nouveaux concepts de haut niveau puis hériter de leurs propriétés. A titre d'exemple, on peut définir le concept de haut niveau « ressource » qui sera hérité par la suite par des concepts de ressources spécifiques aux terminaux ou aux équipements réseaux.

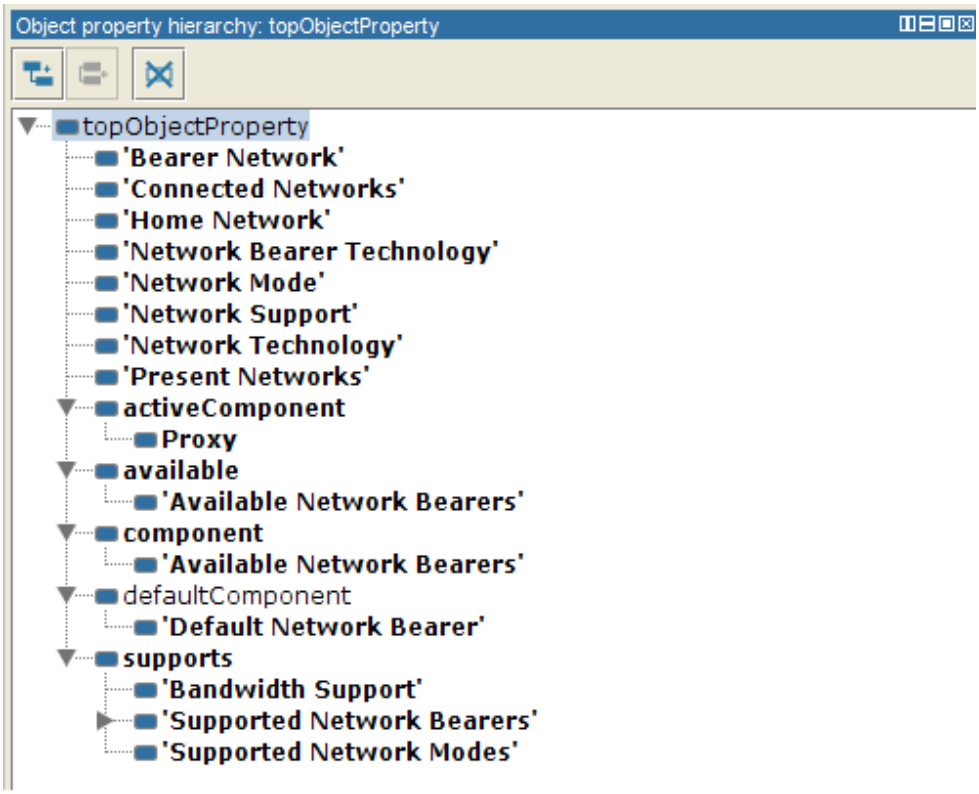


Figure 8 : Exemples de propriétés objets de la classe réseau de *Ubiquity-Ont*

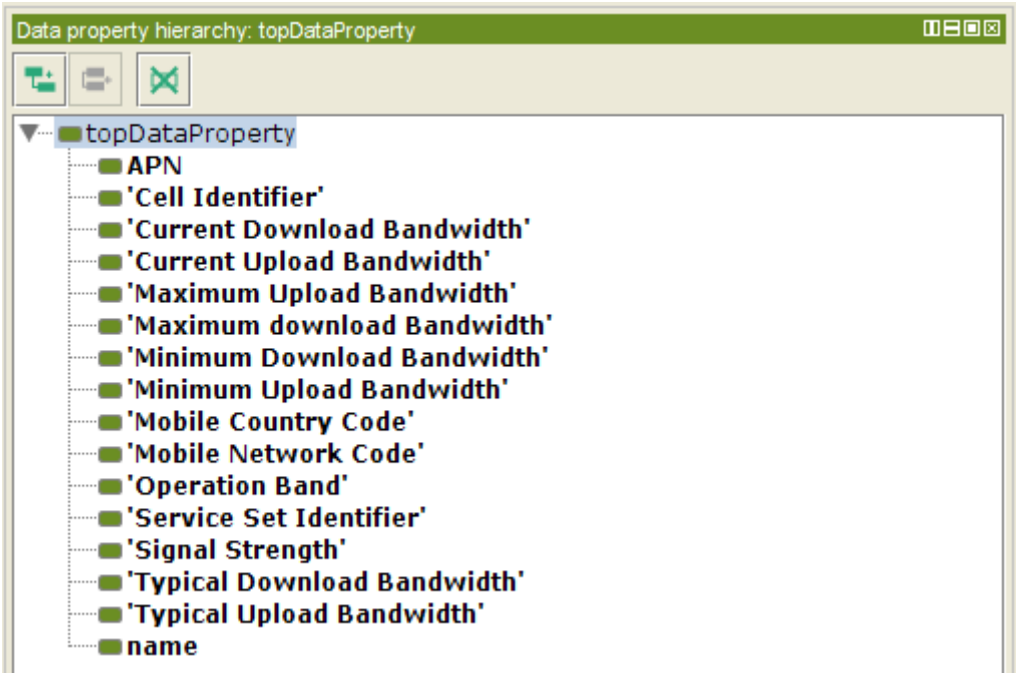


Figure 9 : Exemples des attributs de la classe réseaux de *Ubiquity-Ont*

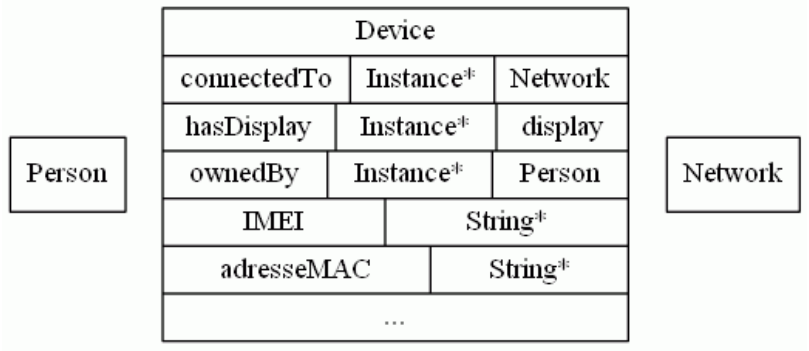


Figure 10 : Exemples d'attributs de la classe équipement (device) de *Ubiquity-Ont*

Un autre exemple qui peut nous encourager à créer de nouveau concepts est celui des utilisateurs. Ces derniers peuvent naturellement hériter des propriétés du concept plus générique qui est Personne. Ainsi, une personne physique pourra avoir plusieurs abonnements à plusieurs réseaux, et donc devenir Utilisateur

A partir de l'instant où on envisage introduire quelques concepts plus génériques et que d'autres concepts en héritent quelques propriétés, des relations d'héritage (Figure

11**Erreur ! Source du renvoi introuvable.**) ou d'autres types de relations puissent apparaitre et qu'on doit définir (voir la section suivante). Mais les relations que nous avons définies pour *Ubiquity-Ont* seront présentées dans ce qui suit.

L'ensemble des relations entre les différents concepts identifiés, ainsi que les détails de ces concepts seront décrits dans la sixième étape.

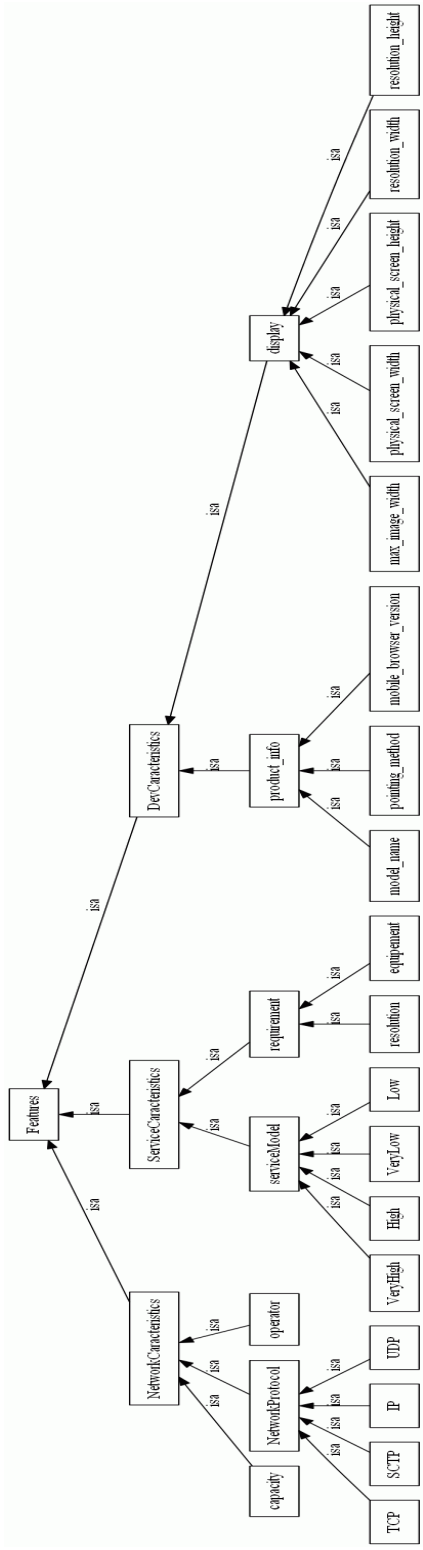


Figure 11 : Exemples de relations d'héritage entre les concepts de *Ubiquity-Ont*

II.7. Sixième étape : Définir les relations entre les concepts

La sémantique qu'on peut introduire avec l'expressivité des langages des ontologies se traduit par la définition d'un ensemble de relations entre les concepts, des propriétés de ces derniers ainsi qu'un ensemble de restrictions et d'axiomes.

La plus importante relation que nous avons utilisée dans l'ontologie *Ubiquity-Ont* est la relation « est-un » (en anglais « is-a »). Avec cette relation on pourra définir des concepts à partir d'autres concepts déjà existants e.g. un équipement « est-une » ressource (c.f. Figure 12 et Figure 13).

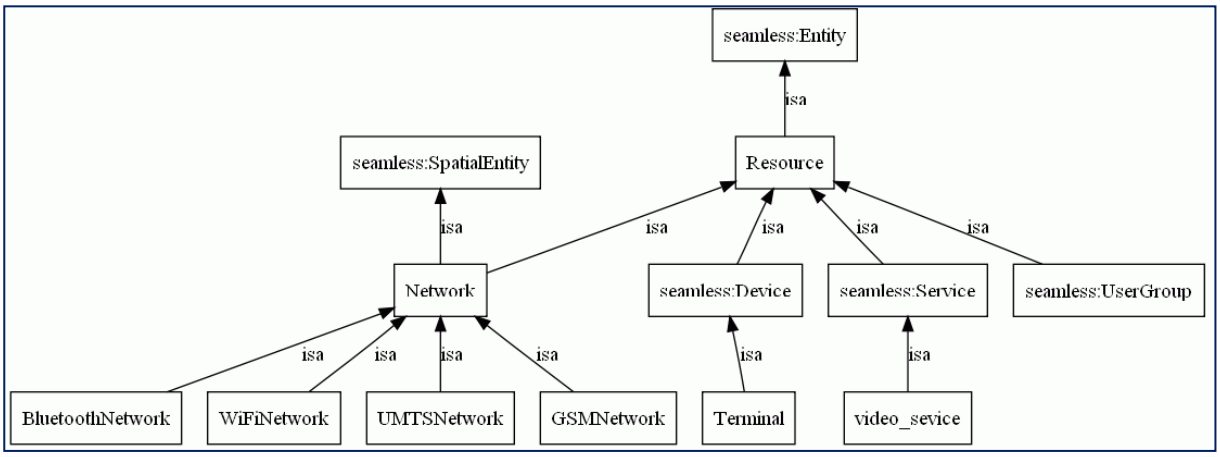


Figure 12 : Exemples de relations « est-un » dans *Ubiquity-Ont*

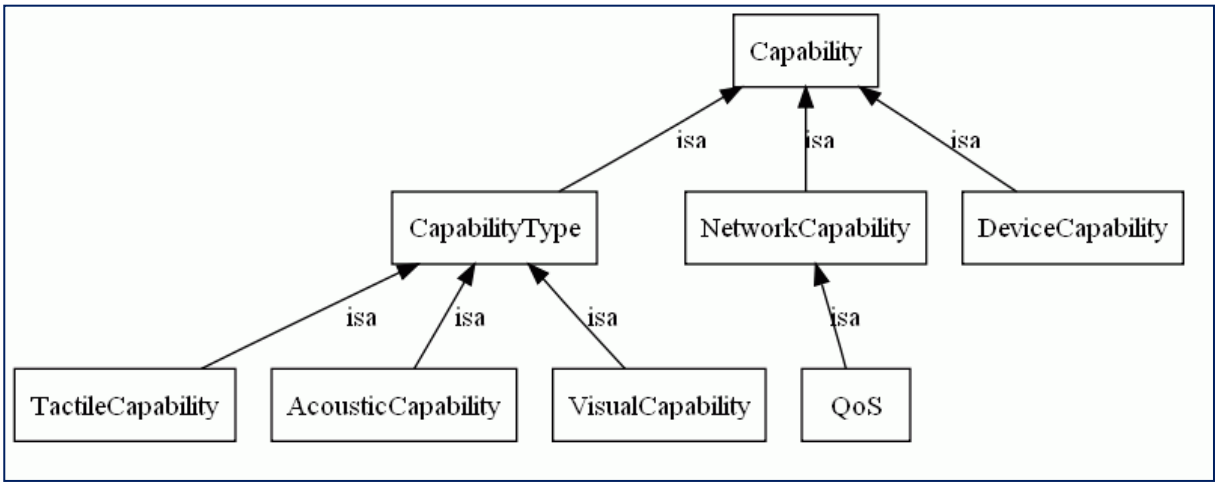


Figure 13 : Exemples de relations « est-un » dans *Ubiquity-Ont*

Un autre aspect important c'est qu'avec les ontologies on pourra définir des propriétés à partir d'autres en utilisant des restrictions. Par exemple :

- une propriété peut être l'inverse d'une autre
- deux propriétés peuvent être symétriques.

Voici un exemple de relations entre les concepts que nous avons défini dans l'ontologie Ubiquity-Ont :

- « utilise » : e.g un utilisateur utilise un terminal
- « est utilisé par » : c'est la propriété inverse de celle « utilise »
- « a un abonnement à » : e.g. un utilisateur a un abonnement à un réseau donné
- « est connecté à » : e.g. un terminal est connecté à un réseau
- « fournit l'accès à » : e.g. un réseau fournit l'accès à un terminal donné (inverse de « est connecté à »)
- « Contient la ressource» : e.g. un réseau contient certaines ressources
- « est une ressource de » : e.g. une ressource appartient à un réseau donné (inverse de « contient la ressource »)
- « a la capacité équipement » : e.g. un équipement a certaines capacités
- « a la capacité réseau » : e.g. un réseau a certaines capacités
- « est le propriétaire » : e.g. une personne est propriétaire d'un terminal

Les figures (Figure 14, Figure 15, Figure 16) illustrent des exemples de ces concepts et relations dans l'ontologie *Ubiquity-Ont* proposée, par exemple un équipement donné est sous la propriété d'un utilisateur donnée, cet équipement est connecté à un réseau donné.

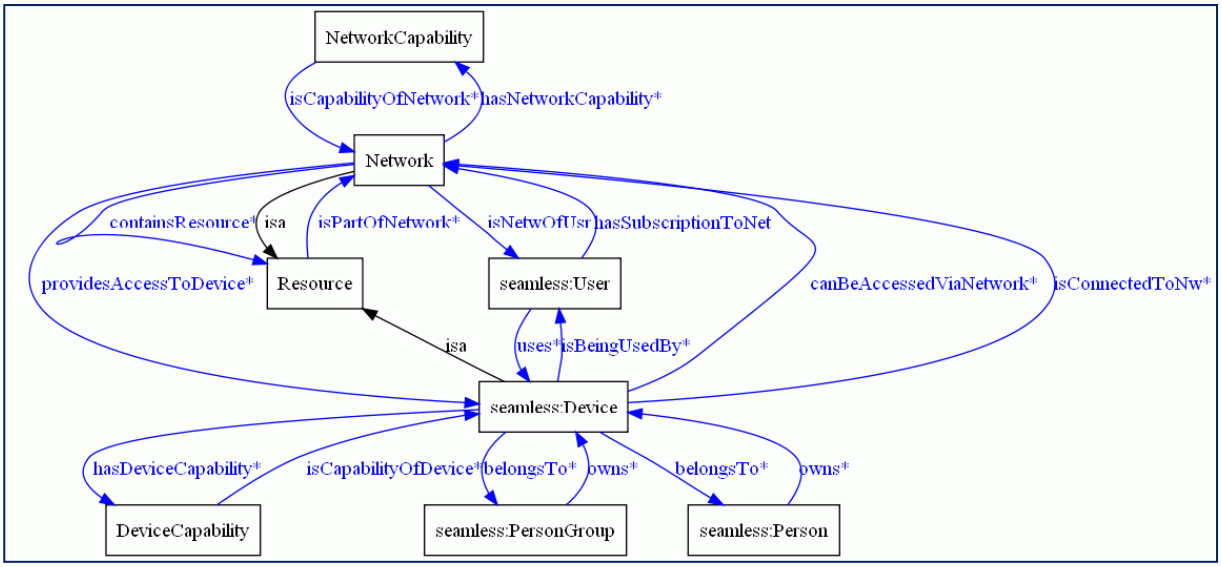


Figure 14 : Exemples de relations sémantiques entre les concepts dans *Ubiquity-Ont*

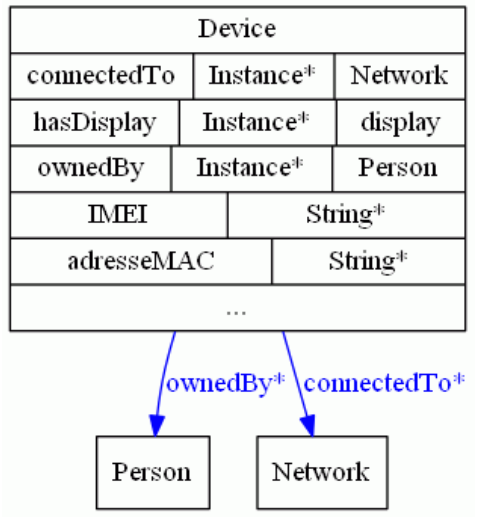


Figure 15 : Exemples de relations entre les concepts dans *Ubiquity-Ont*

Après avoir défini les concepts et les relations entre eux, il faut enrichir *Ubiquity-Ont* par les propriétés de chaque concept. Les propriétés d'un concept décrivent son contexte. Ainsi, un concept dans l'ontologie peut être présenté comme étant une classe (analogie avec le langage de modélisation universel UML) avec ses propriétés, et si on veut présenter

graphiquement cette « classe » on aura une fiche contenant l'ensemble des propriétés portant le nom de cette classe (voir Annexe A).

En pratique il faut bien énumérer les propriétés des entités du système (paramètres qui contiendront les informations du contexte dans une instance donnée) afin de bien décrire leurs contextes.

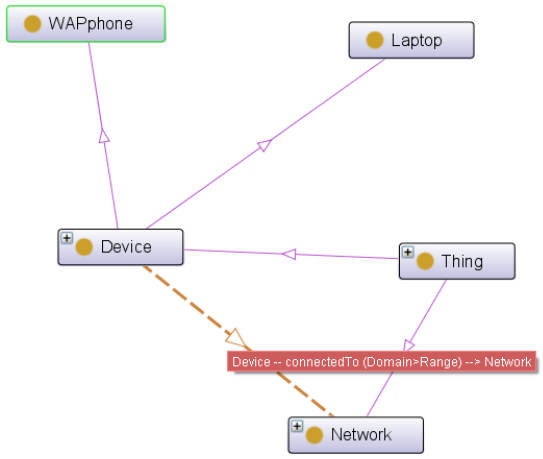


Figure 16 : Exemple de relation entre les concepts « équipement » et « réseaux »

Pendant la conception nous avons énuméré les entités et leurs propriétés de l'environnement d'une façon générique, mais cette liste sera complétée et enrichie avec les spécificités de chaque cas d'études qu'on peut traiter (c.f. Chapitre 5). En effet, l'énumération exhaustive de ces propriétés nécessite un effort au niveau de chaque partie du système ; en d'autres termes, si le gestionnaire de mobilité a besoin de certaines informations de contexte pour la prise de décision de mobilité c'est à partir des scénarios d'utilisation, des algorithmes implémentant l'intelligence de prise de décision de mobilité et des technologies à utiliser dans ces scénarios qu'on pourra énumérer la liste complète de ces propriétés. La Figure 17 présente les trois concepts utilisateur, équipement et réseau, leurs propriétés et les relations.

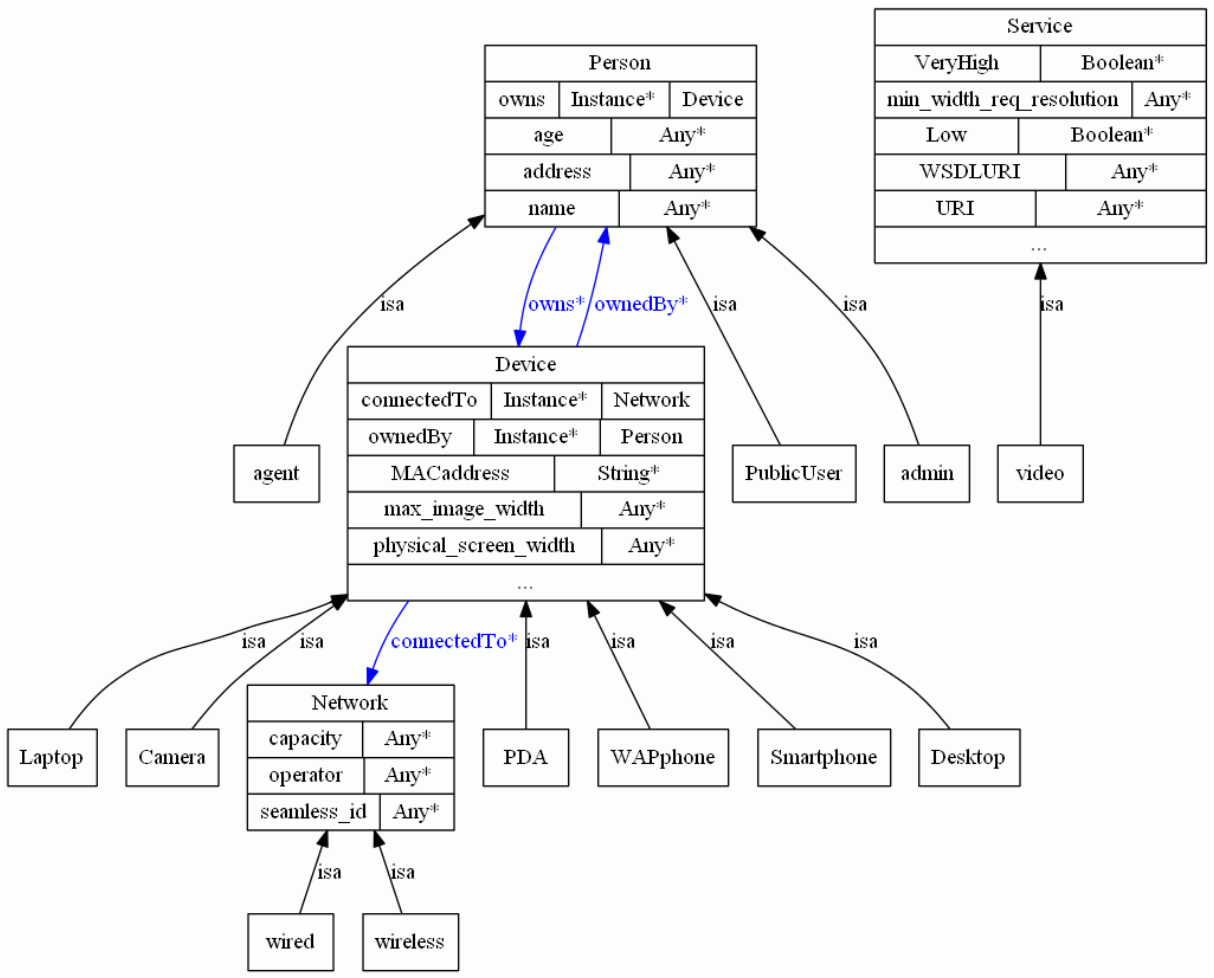


Figure 17 : Exemple d'attributs et de relations entre classes

II.7.1. Cardinalité des attributs

La cardinalité des attributs définit le nombre de valeurs qu'un attribut peut avoir. Certains systèmes distinguent entre cardinalité unique (n'autorisant qu'une seule valeur) et cardinalité multiple (autorisant une ou plusieurs valeurs). Un « *utilisateur* » peut disposer d'un ou plusieurs équipements de type « *Smartphone* », donc c'est une cardinalité multiple. Par contre, un « *Smartphone* » ne peut avoir qu'un seul propriétaire (*utilisateur*), donc une cardinalité unique.

Certains systèmes permettent de spécifier une cardinalité minimale et maximale pour décrire plus précisément le nombre de valeurs d'un attribut. Cardinalité minimale N veut

dire qu'un attribut doit avoir au moins N valeurs. Cardinalité maximale M veut dire qu'un attribut peut avoir au maximum M valeurs. Par exemple dans Ubiquity-Ont, et pour définir un réseau donné ayant un débit de transmission maximal cette valeur est bien évidement unique et exacte, donc on limite la cardinalité de la valeur de l'attribut définissant cette caractéristique à « exactement égal à 1 » (c.f. **Erreur ! Source du renvoi introuvable.**).

Figure 18 : Exemple de définition des cardinalités des attributs

II.7.2. Type de valeur des attributs

La facette type de valeur décrit les types de valeurs pouvant être affectés à l'attribut. Voici une liste des types de valeurs les plus typiques :

- **Chaîne de caractères** est le type de valeur le plus simple utilisé pour des attributs tels que *nom* : la valeur est une simple chaîne de caractères
- **Nombre** (quelquefois des types de valeurs Enveloppe et Entier plus spécifiques sont utilisés) décrit des attributs ayant des valeurs numériques. Par exemple,

A titre d'exemple dans la **Erreur ! Source du renvoi introuvable.**, la valeur de la version u son nom du standard d'un réseau donné prend comme type : « chaine de caractère » d'une cardinalité maximale égale à 1.

Les figures (Figure 19 et Figure 20) illustrent quelques attributs de certains concepts/classes avec la spécification des types de leurs valeurs. Par exemple la valeur de « requireReso »

l'attribut qui décrit la résolution requise pour un service donnée est de type « nombre entier » (Integer) ou bien pour décrire les ressources d'une entité (réseau par exemple) est de type « chaîne de caractères ».

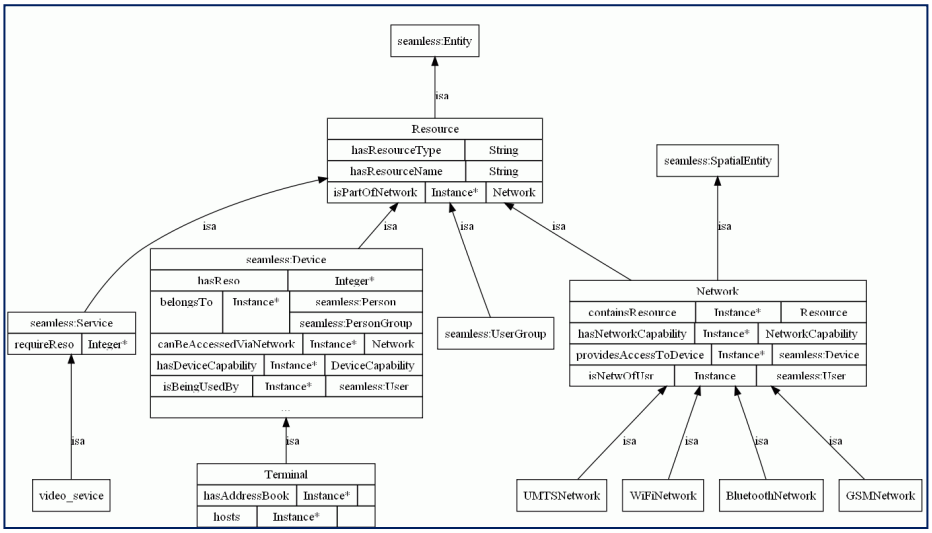


Figure 19 : Exemples des propriétés des concepts

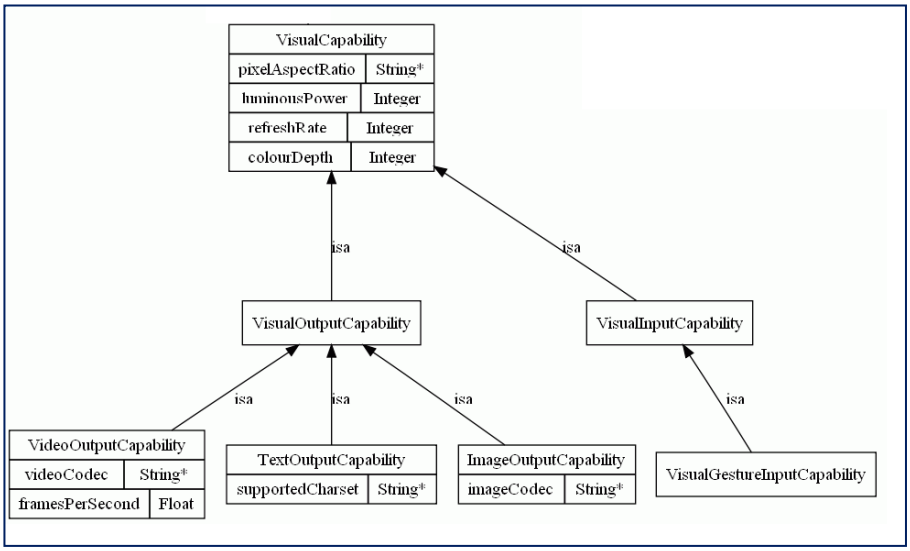


Figure 20 : Exemples des propriétés des concepts

II.8. Septième étape : créer les instances

L'ontologie ainsi décrit chaque concept dans « sa fiche », c'est la première partie du gestionnaire du contexte décrite dans le paragraphe II.2 de ce chapitre qui contient les informations sans sémantique, mais contient une sémantique traduite par les relations entre ces concepts.

Cette description « riche » comporte ainsi deux niveaux de description : la description formelle des concepts et des relations, nommé les Tbox et la description des instances de ces concepts, l'ensemble de ces instances présentent les Abox de l'ontologie.

Une instance décrit une entité physique. La Figure 21 présente des exemples d'instances du concept Smartphone en relations (« owned-by » et réciproquement « owns ») avec les instances du concept Utilisateur. Dans cette figure on peut facilement remarquer les relations entre les concepts et les mêmes relations avec les instances de ces concepts.

A titre d'exemple, une instance du concept utilisateur pourra être l'utilisateur qui a comme prénom (son identifiant par exemple) « Bob », cet utilisateur utilise le terminal qui a comme identifiant « MC35_1 » (un PDA MOTOROLA de modèle MC35) qui est une instance du concept terminal. La Figure 22**Erreur ! Source du renvoi introuvable.** présente cet exemple. Aussi la Figure 23 présente un exemple d'instance du concept service. On voit bien les attributs de ce cette instance (exemple illustré par la figure : le nom du service, son type et la résolution requise, etc.).

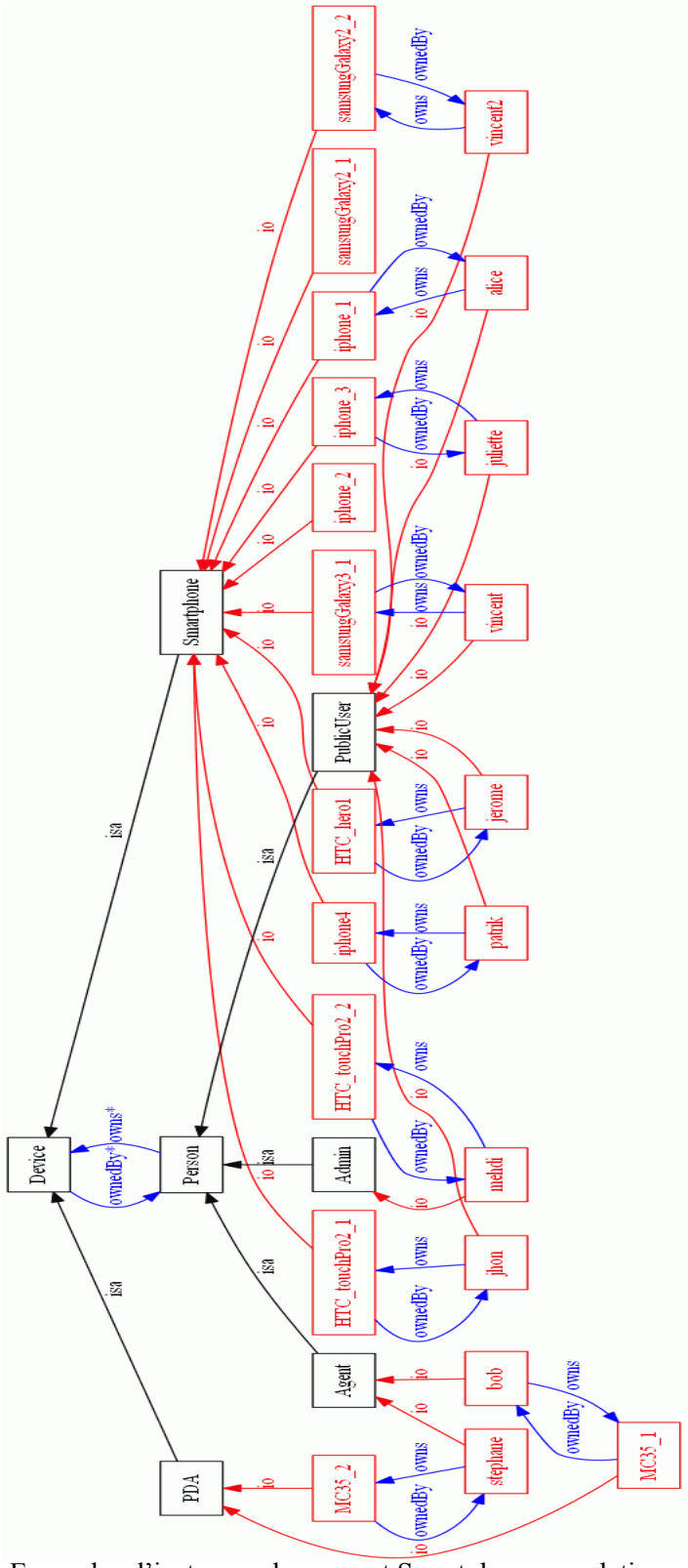


Figure 21 : Exemples d'instances du concept Smartphone en relations avec les instances du concept Utilisateur

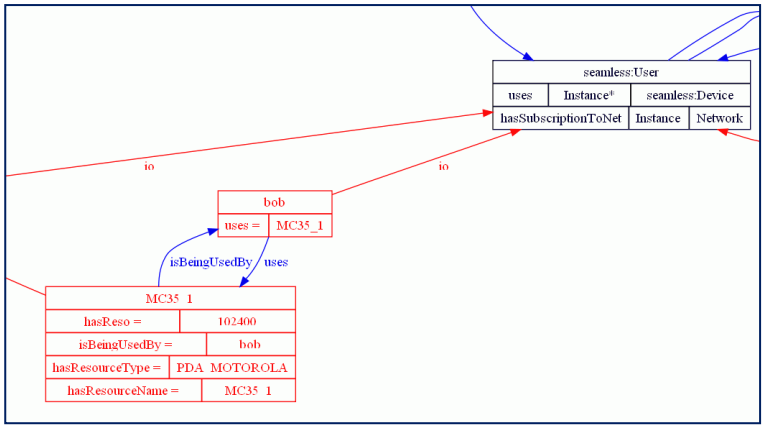


Figure 22 : Exemples d'instances des concepts dans *Ubiquity-Ont*

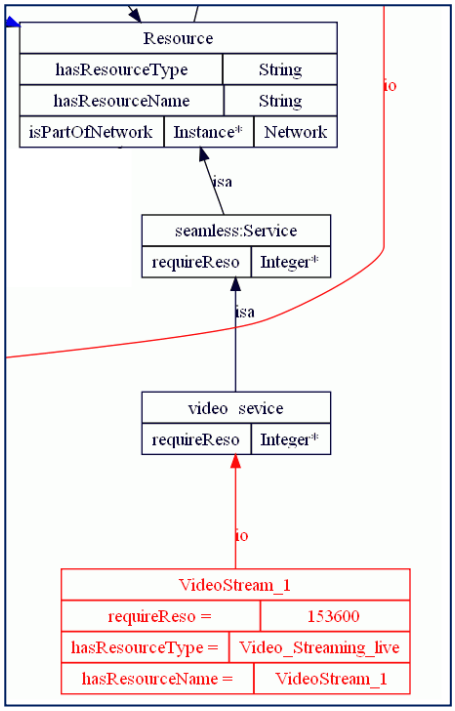


Figure 23 : Exemple d'instance du concept Service dans *Ubiquity-Ont*

III. Outils utilisés pour la conception de l'ontologie

Les éditeurs d'ontologie suivants sont gratuits et téléchargeables.

- « Protégé » est le plus connu et le plus utilisé des éditeurs d'ontologie. Open-source, développé par l'Université de Stanford, a évolué depuis ses premières

versions (Protégé-2000) pour intégrer à partir de 2003 les standards du Web sémantique et notamment OWL. Il offre de nombreux composants optionnels : raisonneurs, interfaces graphiques.[27]

- SWOOP [28]est un éditeur d'ontologie développé par l'Université du Maryland dans le cadre du projet MINDSWAP. Contrairement à « Protégé », il a été développé de façon native sur les standards RDF et OWL, qu'il prend en charge dans leurs différentes syntaxes (pas seulement XML). C'est une application plus légère que « Protégé », moins évoluée en termes d'interface, mais qui intègre aussi des outils de raisonnement.
- KMgen est un éditeur d'ontologie pour le langage KM (The Knowledge Machine).

Avec l'émergence du marché des technologies du Web sémantique, on peut noter l'apparition depuis 2005 d'outils logiciels proposés par des éditeurs commerciaux. On peut en citer:

- SemanticWorks faisant partie de la suite d'outils XML développée par Altova. Il supporte le langage OWL à travers sa syntaxe XML.
- « TopBraid Composer » est développé par TopQuadrant. Son interface et ses fonctionnalités ressemblent beaucoup à celles de « Protégé » (le développeur principal de TopBraid étant l'ancien développeur des extensions OWL de Protégé).
- Ontology Craft Workbench développé par la société Ontologos-corp suite aux travaux de l'équipe Condillac de l'Université de Savoie. Les ontologies sont disponibles aux formats XML et OWL.

Nous avons développé l'ontologie, présentée dans ce chapitre, à l'aide de l'outil « Protégé ».

Les figures (Figure 24 et Figure 25) suivantes illustrent des imprimés écrans d'écran de l'outil Protégé.

[27] En ligne, Protégé. [En ligne] 07 2012. http://protege.stanford.edu/
[28] En ligne, SWOOP. [En ligne] 07 2012. http://www.mindswap.org/2004/SWOOP/

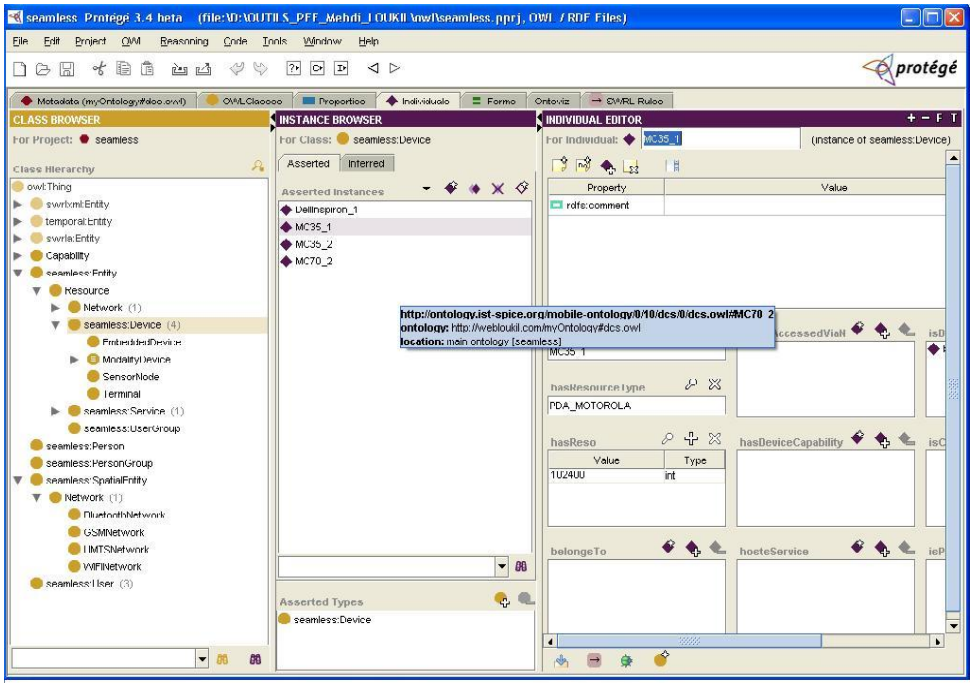


Figure 24 : Une impression d'écran de l'outil Protégé

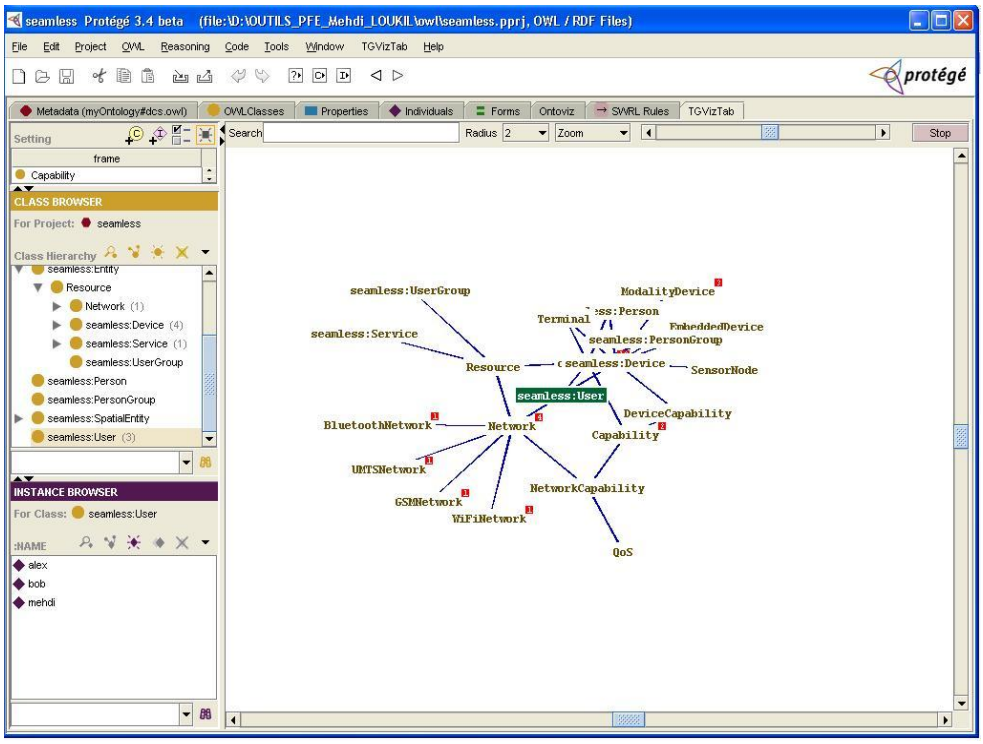


Figure 25 : Exemple de Plugin Protégé permettant de visualiser graphiquement les concepts

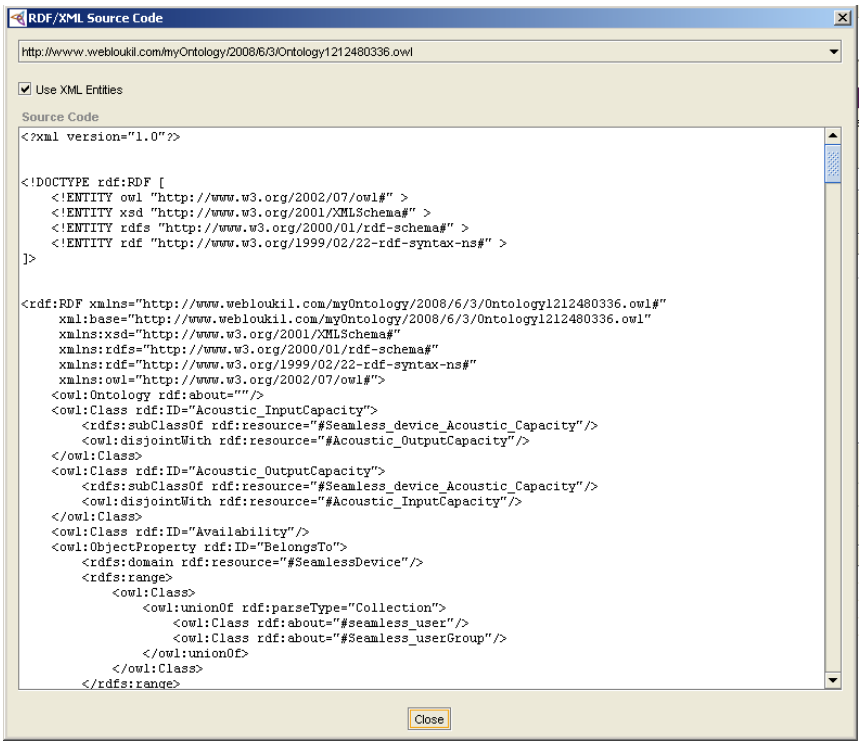


Figure 26 : Extrait du code source OWL/RDF de l'ontologie proposée (voir Annexe 3)

Cet outil offre une interface graphique aisée, des plugins très utiles (voir les figures dans Annexe 1) et des possibilités de corrections automatiques en utilisant des interfaces avec des raisonneurs externes (tel est le cas avec RacerPro).

L'ontologie ainsi se présente comme étant un fichier OWL encapsulé dans RDF puis dans XML (voir Figure 26 et Annexe 1 et Annexe 3).

III.1. Les moteurs d'inférence sur les ontologies :

Un moteur d'inférence (du verbe "inférer" = déduire) est un logiciel correspondant à un algorithme de simulation des raisonnements déductifs [75].

Un moteur d'inférence permet aux systèmes experts de conduire des raisonnements logiques et de dériver des conclusions à partir d'une base de faits et d'une base de connaissances.

Actuellement, il existe plusieurs moteurs d'inférences gratuits ou commerciaux tels que Racer[29], Pellet[30], Fact[31], Fact++[32] et F-OWL[33]. La plupart de ces moteurs est conçue pour raisonner sur les logiques de description, mais acceptent en entrée des fichiers OWL. Certains moteurs d'inférence ne peuvent raisonner qu'au niveau terminologique (c'est-à-dire au niveau des concepts et des propriétés, TBox) alors que des moteurs comme Pellet et Racer permettent de raisonner aussi sur les instances de concepts (ABox).

Par la suite pour exploiter cette ontologie le moteur d'inférence prend comme entrée (input) ce fichier OWL pour offrir une interface permettant d'interroger l'ontologie.

Nous avons utilisé RacerPro version 1.9.2 (beta) avec une licence académique (demandée par l'entreprise vendant ce produit spécialement pour développer notre démonstration de test). Avec cette licence plusieurs fonctionnalités ont été désactivées (entre autres l'envoi distant des requêtes à RacerPro). Pour contourner ce problème nous avons développé un programme qui communique localement avec RacerPro. En fait, ce programme envoi les requêtes (commandes lignes) à RacerPro avec des sockets TCP et reçoit aussi la réponse du moteur d'inférence de la même technique.

IV. Une base de données sémantique

Afin d'évaluer l'ontologie proposée nous avons conçu un gestionnaire de requêtes sémantiques pour pouvoir exploiter la base des connaissances. Ainsi, nous obtenons une base de données sémantique. Cette base de données doit fournir aux composants logiciels « consommateurs du contexte » la bonne information. Le gestionnaire de requêtes sémantiques doit interpréter les requêtes qu'il reçoit, puis extraire la réponse à partir de la base des instances des concepts de l'ontologie.

[29] En ligne, RACER. [En ligne] http://www.racer-systems.com/
[30] En ligne, Pellet. [En ligne] 03 2008. http://www.algo.be/ref-projects.htm#KMgen
[31] En ligne, FaCT. [En ligne] 06 2008. http://www.cs.man.ac.uk/~horrocks/FaCT/
[32] En ligne, FaCT++. [En ligne] 03 2008. http://owl.man.ac.uk/factplusplus/
[33] En ligne, F-OWL. [En ligne] 05 2008. http://fowl.sourceforge.net/index.html

La conception fonctionnelle de ce gestionnaire de requêtes est schématisée sur la Figure 27. Ce gestionnaire de base de données a été conçu par blocks fonctionnels essentiels puis nous l'avons optimisé afin de réduire les temps de réponse.

Les principaux blocs fonctionnels sont :

- Le formulateur de requêtes « query formulator » : il réécrit la requête reçue avec le langage spécifique du moteur d'inférence utilisé.
- Le moteur d'inférence « inference engine » : c'est le composant essentiel pour identifier les branches concernées par la requête Tbox (concepts et relations) de l'ontologie.
- Le chargeur du Tbox « Ontology charger : est un composant lié au moteur d'inférence. Il charge le Tbox concerné par la requête.
- L'exécuteur de requête « query executor » : A partir du Tbox il charge le Abox correspondant. Ensuite, ce bloc exécute la requête.
- Le formulateur du résultat : à partir du résultat de la requête, ce composant réécrit le résultat pour le fournir au composant qui a lancé la requête.

Nous avons développé une plateforme de teste selon cette conception. Nous avons utilisé JAVA comme langage de développement. Nous avons intégré Pellet comme moteur d'inférence tout en utilisant l'API de développement Jena pour pouvoir l'intégrer à notre gestionnaire de requête. L'ontologie *Ubiquity-Ont* développée avec Protégée 3.5 et le langage OWL-DL/RDF. Le gestionnaire de la base de données sémantique est intégré dans une machine à état d'un agent logiciel développé sur la plateforme JADE tout en utilisant certaines APIs spécifiques à savoir OWL-API. Nous avons testé le fonctionnement avec des requêtes simples et d'autres complexes. Le formulateur de requête réécrit les requêtes au format SPARQL (SPARQL Protocol and RDF Query Language). Les requêtes simples consistent à solliciter un simple attribut d'une entité donnée. A titre d'exemple : « qu'elle est l'adresse postale de l'utilisateur nommé Bob ». Les requêtes complexes consistent à solliciter plusieurs attributs ou même entités ou relations entre entités tout en fournissant un minimum d'informations. Par exemple, solliciter les exigences d'un service donné en termes de QoS tout en fournissant l'identifiant ou le nom de l'utilisateur de l'équipement qui exécute ce service.

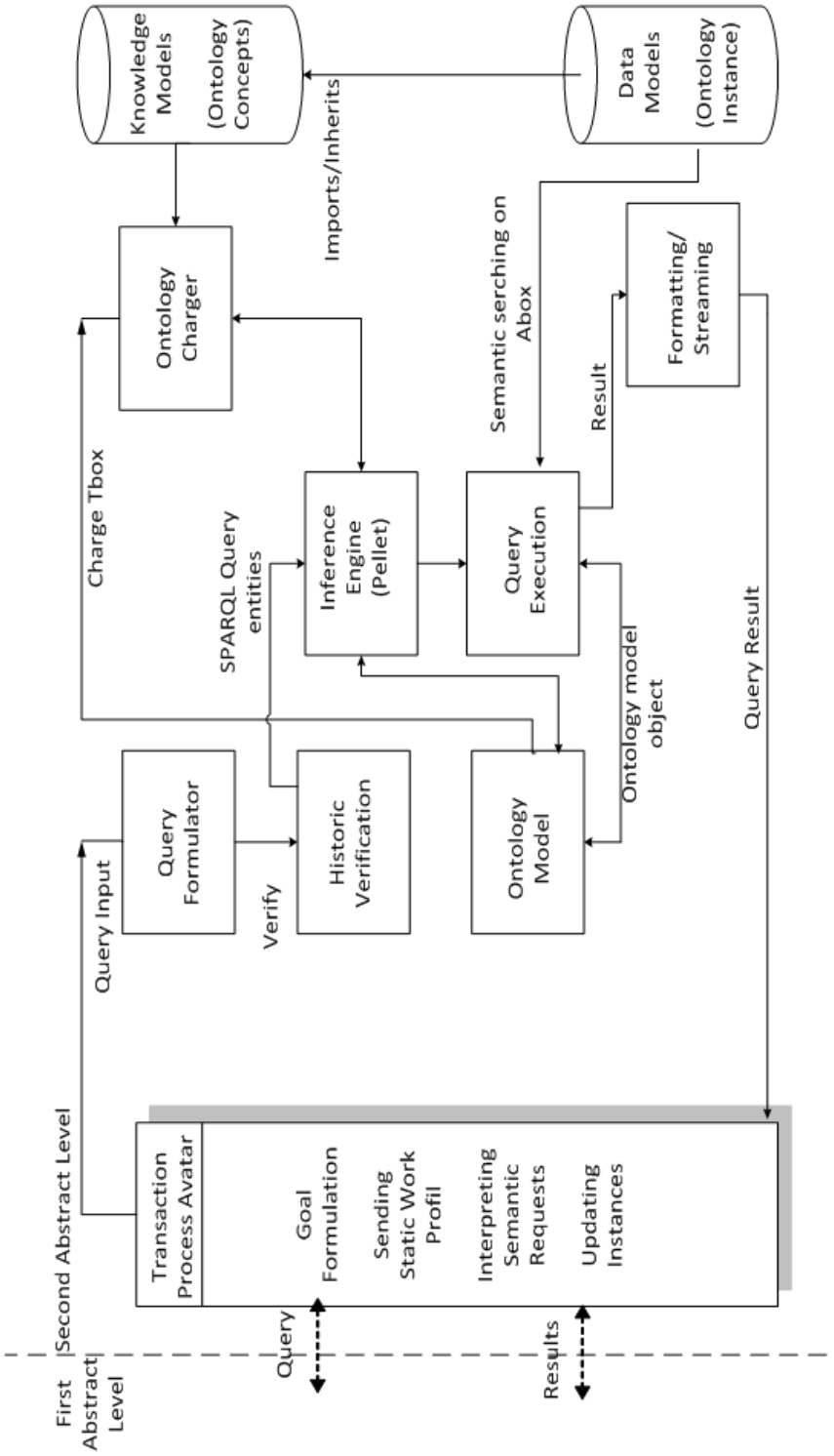


Figure 27 : Gestionnaire de base de données sémantique

Nous avons peuplé l'ontologie par des instances tout en optant plusieurs scénarios et tout en augmentant à chaque fois le nombre d'instances. Pour des raisons de simulations et tests de la plateforme nous avons fait varier le nombre d'instances d'une dizaine à plusieurs milliers (nous avons atteint les 30000 instances).

Les résultats de cette expérimentation ont montré que *Ubiquity-Ont* peut être utilisée comme base de connaissances des informations contextuelles dans un environnement centré sur l'utilisateur et contenant plusieurs technologies réseaux, équipements et services hétérogènes. Un autre résultat important est qu'avec le gestionnaire de requêtes proposé, nous avons réussis à exécuter des requêtes écrites dans un langage différent de celui utilisé par les moteurs d'inférence standardisés et répandus. Ceci a été possible grâce à l'utilisation du bloc « query formulator » qui réécrit les requêtes tout en utilisant un petit dictionnaire de translation de termes. Ce composant nécessite encore du travail de recherche et d'optimisations. Nous laissons ce travail comme perspectives. Les temps de réponse aux requêtes obtenus sont encourageants. A titre d'exemple pour 20000 instances le temps de réponse pour une requête complexe (ex. « quelles sont les classes accessibles du service vidéo streaming N qu'on peut exécuter sur l'équipement de l'utilisateur Bob ? ») était seulement de 318 ms. Cette première expérimentation nous a encouragé à encore utiliser notre base de données sémantique pour d'autres cas d'utilisation et de l'optimiser pour l'intégrer dans une architecture plus complète. Nous présentons nos contributions pour ces axes dans les prochains chapitres.

Dans une deuxième phase, nous avons optimisé le gestionnaire de base de données sémantique par l'ajout de certains blocs supplémentaires qui accélèrent l'accès à l'information. L'idée principale de cette optimisation consiste à sauvegarder l'historique des branches de l'ontologie résultats des requêtes. En effet, nous avons constaté que l'extraction du modèle Tbox concerné par la requête prend plus que 80% du temps total d'exécution de la requête. Ces blocks sont :

- La vérification de l'historique : ce bloc vérifie si la requête a déjà été exécutée dans le passé. Les requêtes sont ainsi enregistrées et identifiables par des nominations sémantiques (le nom d'une requête déjà enregistrée comprend les mots clés essentiels pour l'exécution de cette dernière à savoir les concepts concernés, les attributs/concepts sollicités ainsi que les relations entre ces derniers). Un compteur

de nombre de requêtes est utilisé pour gérer la durée de vie de l'historique pour chaque requête. En effet, si une requête donnée est très sollicitée d'une façon répétitive, le gestionnaire la sauvegarde plus longtemps dans son historique, contrairement aux requêtes moins exécutées dont l'historique n'est pas gardée. Si ce bloc trouve la requête dans l'historique il invoque un second bloc pour charger le Abox correspondant (sans même passer par le chargement du modèle de l'ontologie correspondant, Tbox).

- La sauvegarde des branches de l'ontologie correspondantes aux requêtes sauvegardées.

Avec ces optimisations nous avons pu optimiser les temps de réponses aux requêtes sémantiques. En effet, nous avons pu réduire les temps de réponse d'un facteur de l'ordre de 3. Par exemple pour une base de connaissances comportant 20000 instances et une requête complexe similaire à celle exécutée avant les optimisations, nous obtenons un temps de réponses d'environ 111 ms (318 ms avant optimisations). Pour 30000 instances nous obtenons des temps de réponses d'environ 150 ms comme moyenne pour des requêtes similaires de types complexe. La Figure 28 Figure 28 : Temps de réponse des exécutions des requêtes sémantiques illustre l'évolution du temps de réponses des exécutions des requêtes de type complexe en fonction du nombre des instances dans l'ontologie. Nous constatons que l'évolution est une fonction presque linéaire affine de coefficient faible presque égal à 0,00516. L'évolution du temps des réponses du gestionnaire de notre base de données sémantique est ainsi faible par rapport à l'évolution du nombre des instances dans cette base.

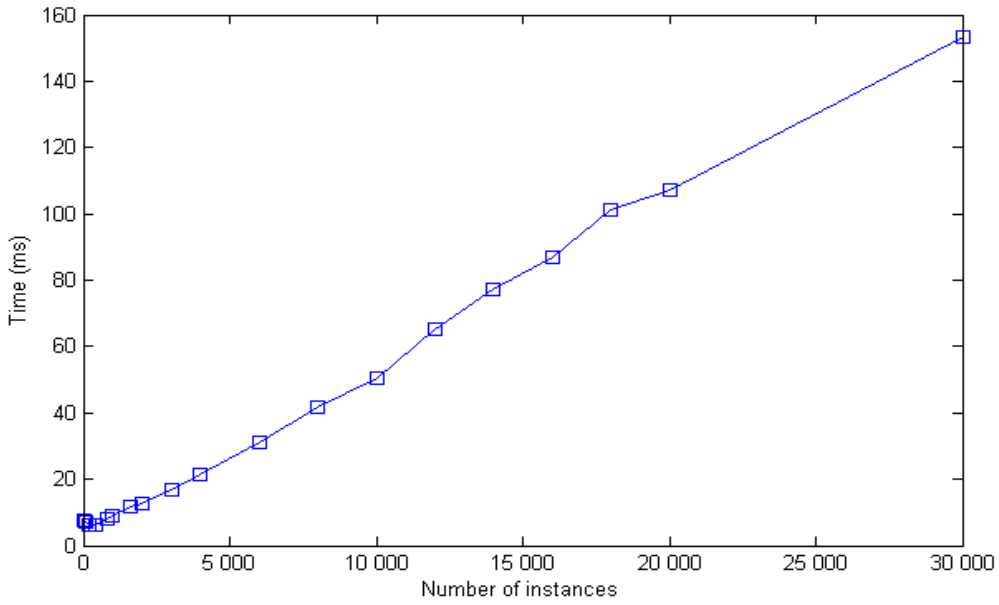


Figure 28 : Temps de réponse des exécutions des requêtes sémantiques complexes

Conclusions

La disponibilité d'une ontologie pour le domaine des TIC semble devenir une nécessité afin d'accompagner les nouvelles tendances vers l'informatique ubiquitaire et les réseaux ambiants. Elle permettra de représenter l'environnement technologique par des bases de données sémantiques sur lesquelles il sera possible de développer des raisonneurs automatisés pour la prise de décision dans les systèmes autonomiques.

La construction d'une ontologie est loin d'être une tâche facile. Elle demande un travail approfondi d'analyse et de compréhension du domaine et des utilisateurs du domaine. La construction d'une ontologie générique et extensible est donc un travail qui nécessite certainement encore beaucoup de temps. Par exemple, le projet « Cyc » qui a pour objectif de proposer une ontologie désignée par « ***cyc d'encyclopedia*** » (prononcé *saïk*), a été lancé depuis 1984, et continue encore à évoluer.

De plus, bien que la conception des ontologies doit être la plus objective possible, elle est néanmoins affectée par la subjectivité du concepteur et des informations. Ainsi deux ontologies d'un même domaine, définies par deux concepteurs différents, présenteront très

certainement des décompositions différentes. Il est donc probable de voir l'apparition de plus en plus d'ontologies différentes, concernant des domaines proches ou identiques, chacune apportant une spécificité ou un point de vue particulier.

Généralement, on ne vise pas à développer une ontologie complète pour un domaine. Développer une ontologie consiste à définir un ensemble de données et les structurer afin qu'elles soient utilisées par d'autres programmes et applications logicielles. Par exemple, dans le cadre de nos travaux nous avons développé une ontologie sur les environnements hétérogènes des TIC. Cette ontologie peut être utilisée comme base de données pour toute une série d'applications et de services visant à gérer et ou améliorer le fonctionnement de cet environnement hétérogène.

Dans ce chapitre nous avons introduit et définit, dans sa première partie, la sémantique ses origines et significations. Par la suite nous avons définit l'ontologie Ubiquity-Ont pour décrire les environnements TIC hétérogènes. Pour cela, nous avons proposé une méthodologie que nous avons suivie pour concevoir notre ontologie.

L'ontologie a ensuite été testée sur une mini-plateforme et les résultats de performances ont démontré que son utilisation n'engendre pas des temps de réponse pouvant affecter négativement les services.

L'ontologie proposée est à disposition de la communauté de recherche et accessible sur internet (voir Annexe 3).

Dans la suite de ce mémoire nous présenterons notre architecture overlay permettant une meilleure exploitation de l'ontologie sur des environnements réseaux hétérogènes. Cette architecture est principalement basée sur la gestion du contexte et intègre une base de données sémantique ainsi que deux niveaux de virtualisation.

.

Chapitre 4 : Une Architecture pour la Gestion du Contexte dans les Réseaux

I. Introduction

Les outils de gestion de contexte par les ontologies nécessitent encore une architecture logicielle afin de pouvoir les intégrer et les exploiter dans différents domaines. Cette architecture doit permettre l'interaction des différents composants de l'environnement afin d'échanger les informations de contexte et de les exploiter pour les prises de décision telles que les adaptations de service et la reconfiguration des équipements.

Les composants de l'environnement peuvent être très hétérogènes en termes de technologie, de caractéristiques et de capacités. L'intégration des éléments nécessaires à la gestion de contexte sur ces composants n'est pas toujours faisable et peut avoir des conséquences néfastes sur les performances.

Comme analysé dans l'état de l'art (c.f. Chapitre 3), les solutions existantes pour la gestion de contexte ne sont pas très adaptées à des systèmes étendus, ouverts et hétérogènes. En effet, nous définissons le contexte d'une application comme l'ensemble des informations utiles à son fonctionnement dans un environnement intelligent, et par extension le contexte d'un capteur ou d'un service producteur d'informations de contexte contient les informations qu'il produit caractérisant l'environnement. Par ailleurs, nous souhaitons donner la possibilité aux concepteurs de ces applications et de ces dispositifs de modéliser le contexte selon leur propre vision du monde. Nous avons vu que modéliser le contexte en utilisant des technologies du web sémantique peut offrir cette possibilité.

Ce chapitre, a pour objectif d'apporter une solution architecturale, capable d'intégrer à l'environnement considéré, les éléments nécessaires à la gestion de contexte (ontologie et gestionnaire de contexte, algorithmes de raisonnement et de prise de décision) donnant à l'environnement les capacités des systèmes pervasifs.

Dans le but d'offrir une infrastructure de gestion d'informations de contexte pour les environnements d'intelligence ambiante, il faut la concevoir pour qu'elle fonctionne dans un

monde ouvert [76]. Il est nécessaire de construire cette infrastructure pour qu'elle soit dynamique, qu'elle supporte l'hétérogénéité des dispositifs et des applications, l'extensibilité des modèles de contexte et ne demande qu'un effort minimum aux concepteurs pour s'y rattacher.

Nous allons considérer dans ce chapitre l'environnement des réseaux et des services sans fil et, avec des problématiques d'hétérogénéité entre les technologies, les terminaux et les services.

II. Problématique

Les réseaux de télécommunication ont beaucoup évolué ces dernières décennies. Plusieurs technologies et une multitude de services et d'applications sont disponibles aujourd'hui sur le marché. Les efforts de standardisation et d'optimisation ont été jusqu'ici essentiellement focalisés sur des systèmes mono technologie. Ainsi, chaque système, et plus particulièrement les réseaux opérés, disposent d'outils et de méthodes bien rodés en termes d'opération et de maintenance. Les paramètres et mesures de fonctionnement sont récoltés sans difficultés particulières et les prises de décisions sont effectuées de manière plutôt centralisée.

Aujourd'hui, le paysage des télécommunications est en pleine mutation avec la volonté d'offrir aux utilisateurs le meilleur service possible quel que soit sa position ou son activité. En l'absence d'une technologie unique et efficace pour chaque cas d'usage, la coexistence et la coopération des différents systèmes semblent une bonne solution. Ces systèmes sont hétérogènes de plusieurs façons : technologies, architectures, fonctionnement, services, …

Dans ce cadre, la prise de décision devient très complexe et nécessite de nouveaux outils plus adaptés. En effet, les différentes décisions nécessitent une vision globale qui dépasse le périmètre de chaque réseau ou acteur. Le nombre de paramètres et d'indicateurs à prendre en compte explose et une gestion plus intelligente et plus automatisée nous semble une bonne piste à explorer.

Comment les systèmes de gestion de contexte sémantique, basés sur les ontologies peuvent répondre à ces besoins ? Nous proposons dans ce qui suit, une nouvelle architecture pour un système pervasif, capable d'intégrer les outils et mécanismes proposés dans les chapitres

précédents (ontologie et gestionnaire de base de données sémantique) sur l'environnement existant cité plus haut.

III. Besoins et contraintes

Pour intégrer la gestion de contexte dans l'environnement considéré des réseaux et services hétérogènes, la liste des principaux besoins à considérer est la suivante:

- Interconnecter les sources d'information avec le gestionnaire de contexte. Ceci peut s'effectuer à travers des interfaces et des protocoles d'échange. Proposer des solutions pour générer l'hétérogénéité.
- Proposer une solution alternative pour permettre la récupération des informations de contexte à partir d'entités ne disposant pas de la fonctionnalité de communication avec un gestionnaire de contexte.
- Permettre aux entités de partager les informations de contexte et de pouvoir prendre des décisions intelligentes, afin de tendre vers un système pervasif pouvant avoir une vision globale du système.

Nous nous sommes également fixés plusieurs contraintes afin de proposer une solution universelle pouvant s'adapter facilement à plusieurs domaines.

- Généricité et Interopérabilité
    - L'architecture doit être générique pour pouvoir l'utiliser dans plusieurs domaines;
    - L'architecture doit faciliter la collaboration et l'interfonctionnement de systèmes hétérogènes;
    - L'architecture doit permettre de manipuler différents types de données contextuelles, en tenant compte de leurs degrés de dynamicité.
- Flexibilité, Extensibilité et Scalabilité
    - La solution doit permettre l'extensibilité des systèmes pour pouvoir intégrer de nouvelles entités sans remise en question de l'architecture.

  - La solution doit permettre de considérer des systèmes de très grande taille tels que le domaine des réseaux et services mobiles.
  - La solution doit être flexible pour permettre facilement l'ajout de nouvelles fonctionnalités et de nouveaux mécanismes.

- Faciliter l'automatisation

  - L'architecture doit permettre l'automatisation de certains processus avec un système d'aide à la décision basé sur les informations de contexte. Cette automatisation permettra de minimiser les interventions des opérateurs et des utilisateurs pour configurer leurs services, de plus en plus complexes.

- Amélioration des performances
  - L'architecture ne doit pas affecter négativement les performances du système et de ses entités, essentiellement en termes de délais de réponse et de surcharge.

IV. Architecture de virtualisation à deux niveaux d'abstraction pour la gestion du contexte

Pour répondre aux besoins et contraintes présentés dans la section précédente, notre solution doit :

- Tenir compte des degrés de dynamicité des informations contextuelles : les informations globales et/ou statiques peuvent être gérées à un niveau hiérarchique haut, plus accessible aux différentes entités du système.
- Offrir un niveau conceptuel de haut niveau facilitant le partage entre les entités d'un même système et entre plusieurs systèmes coopératifs.
- Faciliter la compilation des données contextuelles globales et leur historique afin d'en tirer des règles d'optimisation basées sur l'apprentissage
- Rapprocher les traitements sur les informations contextuelles dynamiques des entités réelles de l'environnement et ce pour des raisons de performances.

Nous proposons donc une solution overlay à deux niveaux de virtualisation (c.f. Figure 33). Le niveau le plus haut aura une vue globale et sera responsable de la gestion des informations de contexte statique et des interactions inter-systèmes. Le niveau le plus bas, sera proche des

entités de l'environnement et responsable de la gestion des informations de contexte dynamiques.

Ainsi, l'architecture est à la fois centralisée (du point de vue des utilisateurs) et distribuée sur les réseaux pour offrir de meilleures performances et une meilleure gestion des ressources.

Nous avons également choisi de pousser la virtualisation jusqu'aux entités de l'environnement et ce pour :

- les enrichir de capacités supplémentaires liées à la gestion de son contexte
- les enrichir de capacités supplémentaires pour les prises de décision basées sur le contexte
- leur offrir des moyens uniformisés pour l'échange des données de contexte avec l'environnement
- leur permettre de mieux communiquer entre elles pour une meilleure coopération
- éviter de les surcharger des traitements liés à la gestion de contexte

Nous proposons donc que chaque entité de l'environnement soit représentée au niveau bas de l'architecture par un composant logiciel. L'entité délèguera quasiment tous les traitements liés aux contextes à son représentant. Ces composants logiciels qui vont se substituer aux entités réelles de l'environnement seront désignés par la suite par le terme *Avatars*. La gestion de contexte à ce niveau restera donc distribuée, sur les différents *Avatars* représentant les entités. Le détail de ce niveau sera explicité dans les sections suivantes.

La gestion globale du contexte ainsi que l'orchestration du nuage des *Avatars* seront effectuées sur le niveau le plus haut. Ce niveau offrira donc les outils nécessaires aux *Avatars* pour accéder aux données de contexte global. C'est ce niveau qui sera responsable de l'ontologie permettant d'homogénéiser la totalité du système.

L'architecture proposée sera détaillée dans les sections qui suivent.

V. Premier niveau d'abstraction: une architecture distribuée basée sur des *Avatars*

Ce premier niveau est le plan le plus proche des entités physiques du système. Il est conçu essentiellement pour gérer les informations dynamiques du contexte. Il permet d'offrir aux entités physiques des capacités supplémentaires de stockage, de traitement, de raisonnement et de communication. Les entités physiques, et en premier lieu les terminaux mobiles, se voient donc enrichis de nouvelles capacités leur permettant de mieux interagir dans l'environnement et ses autres composants. Ce premier niveau offre en quelques sortes une « Virtualisation augmentée ».

Chaque entité physique sera représentée par un *Avatar*, qui pourra être implémenté, par exemple, sous forme d'un agent logiciel. Cette approche va non seulement nous permettre d'enrichir les entités par des capacités et des fonctionnalités supplémentaires, mais aussi de leur offrir des moyens supplémentaires et harmonisés en termes d'échange de données et de communication. Autant les entités physiques peuvent être hétérogènes, la notion d'*Avatars* va nous permettre de les harmoniser « virtuellement » dans ce premier plan de l'architecture.

L'idée ici consiste à représenter les entités physiques, logicielles ou juridique par une entité virtuelle et augmentée qu'on appelle « *Avatar* ». Les entités concernées incluent :

- les utilisateurs et les groupes d'utilisateurs (ou communautés),
- les terminaux mobiles,
- les réseaux,
- les équipements et entités des réseaux,
- les services élémentaires et les services composés,
- les différents acteurs des réseaux et des services,
- et toute autre entité communicante qui souhaite intégrer le système, si besoin est.

Un *Avatar* est défini comme un composant logiciel qui s'exécute dans ce premier niveau d'abstraction et capable de communiquer en temps réel avec l'entité qu'il représente, mais aussi avec les autres *Avatars* représentant les autres entités. C'est donc une doublure logicielle de l'entité représentée, enrichie de nouvelles capacités de traitement, de stockage et de communication.

Les principaux objectifs de l'utilisation des *Avatars* sont les suivants :

- Représenter les entités physiques ou logicielles dans l'architecture d'abstraction
- Augmenter « virtuellement » les capacités de traitement et de stockage des entités
- Augmenter les capacités de raisonnement et de prise de décision des entités
- Disposer de nouveaux moyens de communication harmonisés avec les autres entités et/ou les autres *Avatars*
- Décharger les entités représentées de certains traitements afin de préserver leurs ressources

Voici trois exemples démontrant l'utilité de l'introduction de la notion d'*Avatars*.

Exemple 1 : Un *Avatar* représentant un utilisateur peut requêter le niveau le plus haut pour obtenir des informations de contexte concernant un service donné qu'il souhaite invoquer. Cet *Avatar* aura également la possibilité de solliciter des informations de contexte concernant un autre réseau disponible afin de décider s'il souhaite s'y connecter.

Exemple 2 : Un *Avatar* (a) représentant un utilisateur peut communiquer avec les *Avatars* représentant (b) le service à invoquer et (c) le réseau sur lequel l'utilisateur est connecté afin de vérifier la compatibilité du triplet (capacités du terminal, exigences du service, capacités du réseau). S'il y a compatibilité, le service peut être invoqué, dans le cas contraire, l'*Avatar* a pourra solliciter une adaptation du service si disponible, sinon remonter au niveau le plus haut pour demander une recomposition éventuelle du service.

Exemple 3 : les *Avatars* de plusieurs utilisateurs peuvent se constituer en communauté pour une gestion coopérative de la QoE (Quality of Experience) partageant ainsi leurs expériences de QoS pour des couples réseau/service donnés. Cet exemple sera détaillé dans le chapitre 5 faisant partie d'un cas d'étude.

Dans ce qui suit, nous détaillons les aspects fonctionnels et non fonctionnels proposés pour ces *Avatars*.

V.1. Aspects fonctionnels d'un *Avatar*

Un *Avatar* est défini comme un composant autonome qui peut représenter et agir pour et au nom d'une entité physique ou logique du système. C'est un composant logiciel doté d'autonomie et d'intelligence. Il possède une connaissance quasi temps réel sur l'état et le contexte de l'entité qu'il représente.

L'*Avatar* est responsable de plusieurs actions itératives :

- La collecte les informations contextuelles statiques et dynamique avec des cadences différentes et adaptées aux natures des données de contexte concernées. Il a donc une vision complète sur son entité: les informations la décrivant (le contexte statique) ainsi que les informations décrivant son état instantané (contexte dynamique). L'ensemble des informations collectées sera désigné par le *Work Profile* de l'*Avatar*.
- L'analyse et le raisonnement local sur le *Work Profile* pour préparer les prises de décision
- La prise de décision en fonction des contextes actuels de l'entité et de son environnement

Ces trois fonctions sont traduites en trois blocks fonctionnels (c.f. Figure 29).

Ce block fonctionnel, bloc « b » (c.f. Figure 29), est responsable de collecter les informations contextuelles. Il doit bénéficier d'une interface de communication permanente avec l'entité représentée afin de pouvoir mettre à jours régulièrement son *Work Profile*. Cette représentation, formant une base de données propre à l'*Avatar* en question. Elle permet aux autres blocks fonctionnels de l'*Avatar* de trouver rapidement les informations du contexte nécessaires pour résonner dessus rapidement et en temps réel pour prendre les décisions adéquates.

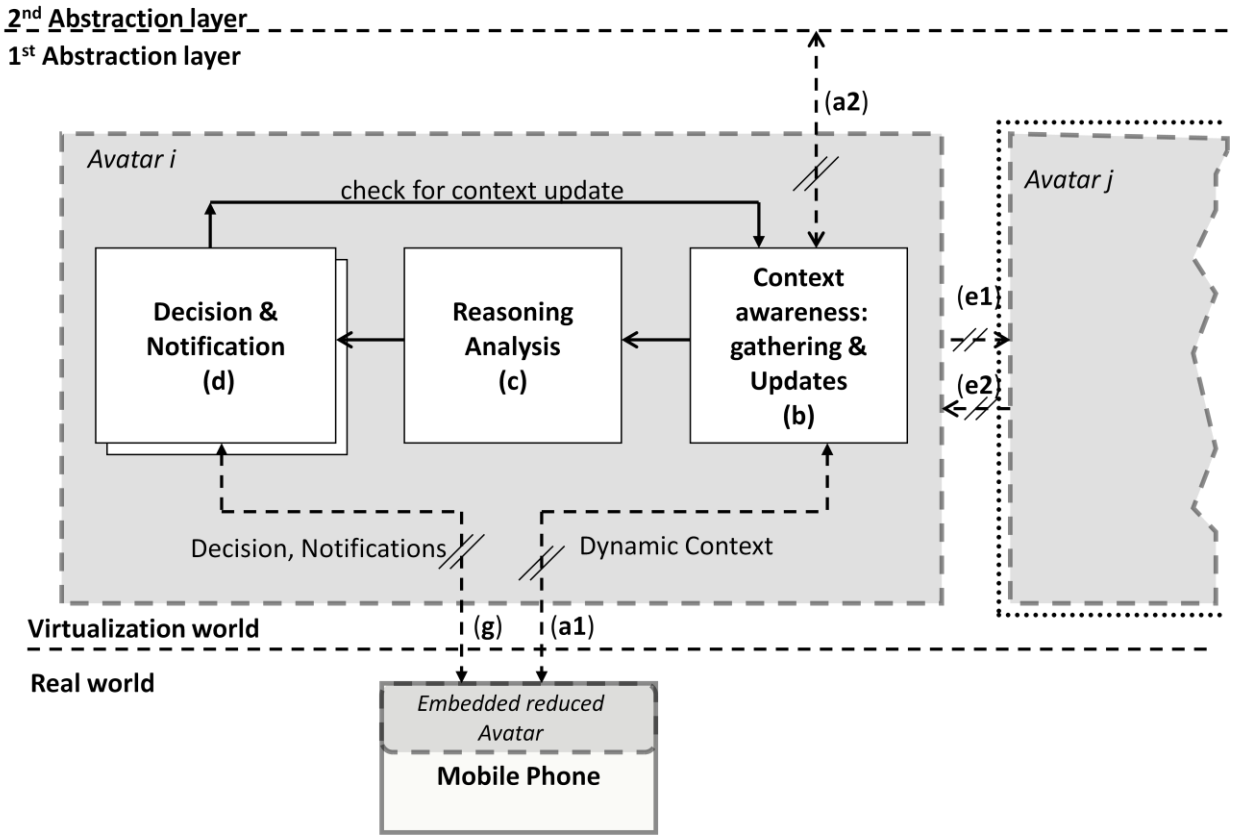


Figure 29 : Blocs fonctionnels d'un *Avatar* générique

V.1.1. Block Fonctionnel « Context Awareness : Gathering and Updates » :

D'un autre côté, l'*Avatar* doit pouvoir communiquer avec le deuxième niveau d'abstraction, c'est à dire le niveau le plus haut. Ceci afin de pouvoir requêter et mettre à jours les informations contextuelles statiques qui concernent l'entité représentée et toute autre entité avec qui il coopère. Ainsi, ce bloc fonctionnel « b » c'est le responsable du maintien des deux interfaces « a1 » et « a2 » (c.f. Interfaces d'échanges).

V.1.2. Block Fonctionnel « Reasoning and Analysis » :

Après la collecte des informations constituant le contexte de l'entité représentée, l'*Avatar* analyse son *Work Profile*, d'une façon régulière (itérative) ou suite à des déclencheurs (triggers). Cette analyse, bloc « c » (c.f. Figure 29) permet de détecter les changements de contexte éventuels pour agir en conséquence selon des algorithmes de décisions spécifiques à chaque entité (c.f. Block « d » Décision & Notification).

Dans certains cas, l'*Avatar* peut avoir plusieurs possibilités d'action, sur différents aspects de l'entité (c.f. Block Decision & Notification ». Il peut y avoir donc plusieurs algorithmes de prise de décision possible. Afin d'éviter les conflits de décision et de choisir le meilleur algorithme en fonction de chaque situation, des règles de priorité sont donc nécessaires. Ces règles sont peuplées au sein de l'*Avatar* au moment de sa création.

C'est en fonction de ces règles que ce Block fonctionnel sollicitera le bon algorithme de décision situé au prochain Block.

V.1.3. Block Fonctionnel « Decision & Notification » :

Ce block fonctionnel peut disposer de plusieurs algorithmes de décision. Par exemple, un *Avatar* représentant un terminal mobile peut véhiculer deux algorithmes de décision, un pour solliciter un handover (mobilité) et un autre pour solliciter une adaptation de service.

Ce block actionnera l'algorithme sélectionné par le précédent Block fonctionnel en fonction de l'analyse du *Work Profile*, prend une décision qu'il notifiera à l'entité représentée.

V.1.4. Block Fonctionnel « Correspondent Entity »

Pour pouvoir communiquer avec son *Avatar*, l'entité représentée doit disposer d'une interface de communication spécifique. Cette interface est une extension fonctionnelle et logique de l'*Avatar*. Elle doit être la plus légère possible en termes de consommation de ressources et d'énergie. Sa principale fonctionnalité se résume à collecter localement et envoyer les données

contextuelles à l'*Avatar* et déclencher l'exécution locale des actions correspondantes aux décisions prises par celui-ci.

V.1.5. Interfaces d'échanges

Un des apports de notre architecture overlay est qu'elle offre aux entités de l'environnement hétérogènes des capacités de communications unifiées. Ces communications et échanges sont assurés par des interfaces que nous devons définir.

Ces interfaces de communication peuvent être assurées par des protocoles existants tel COPS (Common Open Policy Service). Mais afin de maintenir un certain niveau de sémantique, vu qu'on traite des informations de contexte dans un environnement hétérogène, et afin d'alléger les échanges avec les entités physiques représentés par les *Avatars*, nous avons choisi de définir un protocole allégé pour ces échanges, inspiré des concepts utilisés dans les langages de description sémantique tel que XML et RDF et utilisant des couples (attribut, valeur).

L'idée de ce protocole consiste à traduire tout message à échanger entre deux composants de l'architecture sous forme d'une seule chaine de caractères. Cette chaine consiste à des couples « Attribut, Valeur » délimités par des séparateurs (caractères spéciaux non utilisés dans les noms des attributs ni leurs valeurs). Ce protocole semble simpliste, mais avec l'existence de l'ontologie qui définisse les différents concepts, attributs et les types/unités que les valeurs de ces derniers peuvent avoir, il est complet et permet d'échanger tout type d'information entre les composants de notre architecture. Chaque message commence par un mot clés indiquant son type (contexte dynamique, contexte statique, décision, …) et contient d'autres mots clés séparant les sous-types des messages appartenant au même type initial.

La Figure 30 suivante illustre le format générique des messages du protocole d'échange via les interfaces entre les différents composants.

***mot_cles_type_messgae*$*mot_cles_sous_type1*#*valeur_type_1*$*attribut_1*#*valeur*$*attribut_2*#*valeur*$....$*attribut_N*#*valeur*$*mot_cles_sous_type2*#*valeur_type_2*$*attribut_1*#*valeur*$*attribut_2*#*valeur*$....$*attribut_N*#*valeur*$**

Figure 30 : Format générique de messages du protocole d'échange

Ce qui suit nous décrivons les interfaces possibles entre les composants de l'architecture proposée.

L'interface « **a1** » : ce sont l'échange des informations dynamiques entre l'entité représentée et son *Avatar*. L'entité envoi les informations de contexte dynamiques à son Avatar, telles que :

- la liste des réseaux à portée, leurs niveaux de signal, et concernant le réseau dont le mobile est connecté à cet instant : le délai, le jitter, le BER, la BW (celles-ci sont regroupées dans un sous-type d'informations contextuelles concernant les réseaux ainsi dans le message du protocole d'échange nous trouvons un mot clés indiquant ceci avant les couples attribut-valeur) ;
- le service en cours d'utilisation avec les informations leurs caractérisant (le nom ou identifiant du service, son type, l'outil d'exécution, etc.), ainsi un sous-type est déclaré dans le message du protocole d'échange pour séparer ces informations des autres ;
- le niveau actuel de la batterie, l'occupation du CPU et de la mémoire vive, …

L'autre sens d'échange est aussi possible, l'*Avatar* peut envoyer en fait, des informations concernant les services disponibles actuellement dans l'environnement (globalement ou dans la zone concernée par l'entité en question), la liste des réseaux qui ne diffusent pas leurs identités (SSID caché), …

Une phase d'authentification est effectuée au démarrage de la communication avec l'*Avatar* via l'interface « **a1** ». En effet, l'entité envoie à son *Avatar* les informations de son abonnement au système, ce dernier lance une requête d'authentification vers le gestionnaire de contexte globale présent dans le niveau plus haut (second niveau d'abstraction de l'architecture) via l'interface « **a2** ».

La Figure 31 illustre un exemple de message, contenant des informations de contexte dynamique ainsi le mot clés « dynamic_context », envoyé par l'entité représentée (ici c'est l'équipement mobile d'un utilisateur). Ce « dynamic_context » contient les informations (indiquant une idée sur la qualité de la connexion) sur le réseau auquel le mobile est connecté actuellement (Wifi dans l'exemple de la figure).

dynamic_context$type#Network$iface#wlan0$MACaddress#00:0B:FD:EA:88:C9$Channel#1$essid#naxos$delay#130.$jitter#3.8078866$measuredbitrate#2$theoreticalbitrate#0$signalLevel#-76$connection#1$last#0$type#Service$name#email$dataSize#320$userAgent#thunderbird$

Figure 31 : Exemple d'une partie du message « dynamic_context » envoyé par un équipement mobile à son *Avatar*

L'interface « **a2** » : L'avatar communique avec le deuxième niveau d'abstraction de l'architecture via l'interface « **a2** » pour échanger les informations de contexte statiques. Ces informations sont essentiellement des données d'authentification (au début de la session de notre système), des données statiques décrivant le profil utilisateur (profil, préférences et habitudes), celles décrivant les équipements, les services et les réseaux.

Les interfaces « **e1** » et « **e2** » : ces interfaces sont conçues pour les échanges éventuels entre deux Avatars. En effet, comme nous avons indiqué, les Avatars peuvent communiquer entre eux et échanger les informations du contexte pour collaborer pour des meilleures décisions encore sensibles aux contextes. Un exemple de ces collaborations et de ces échanges sera présenté dans le chapitre 5 dans le cadre d'un cas d'étude.

L'interface « **f** » : Tout décision prise par l'*Avatar* sera envoyé via l'interface « f » à l'entité représentée concernée. De l'autre côté, ces messages sont reçus par l'entité représentée et plus particulièrement du block fonctionnel allégé, embarqué dans l'entité représentée. Par la suite la décision prise et envoyée par ce type de message sera exécutée localement. Cette architecture est inspirée du modèle COPS (Common Open Policy Service) connu par PDP/PEP (Policy Decision Points/Policy Enforcement Points).

decision$type#Network$HO#yes$network_dest#wifi$channel#6$essid#orange2$mac#00:00:32:FF:85:84

Figure 32 : Exemple d'un message de notification de décision envoyé via l'interface « f »

V.2. Aspects non fonctionnels

D'autres aspects non fonctionnels sont définis pour les *Avatars*. Ils permettent d'optimiser leurs fonctionnements ainsi que celui de l'architecture globale du système.

En effet, vu que les *Avatar* peuvent représenter des entités mobiles, il paraît utile de toujours les rapprocher physiquement à ces dernières. Ceci permettra entre autre de minimiser les délais d'échange des informations de contexte et des notifications. Nous introduisons donc la mobilité physique des *Avatar*.

D'un autre côté, plusieurs *Avatars* peuvent partager certains aspects ou coopérer fortement autour d'un service ou utilisation donné. Dans ce cas, il serait utile de les organiser en groupes que nous introduisons à travers la notion de mobilité logique.

V.2.1. Mobilité Physique des *Avatars*

En considérant que les *Avatars* vont représenter des entités mobiles (utilisateurs, terminaux des utilisateurs, routeurs mobiles, éléments de services, …), qui peuvent se déplacer géographiquement dans l'environnement, et ainsi peuvent s'éloigner de l'emplacement d'exécution de leurs *Avatars*. En effet, la collecte des informations de contexte nécessite des échanges qui peuvent être assez fréquents en fonction de la dynamicité de l'environnement. De plus, les décisions peuvent être urgentes et ne tolèrent pas beaucoup de délais de transmission pour arriver à l'entité représentée. Ainsi rapprocher l'*Avatar* de l'entité qu'il représente peut donc contribuer à minimiser les délais des échanges à travers les réseaux.

D'un autre côté, les entités de l'environnement se retrouvant à un instant donné à une localisation physique partagent plusieurs informations de contexte spécifique à cet emplacement. Par exemple, les entités physiques des réseaux (équipements, routeurs, points d'accès, BTS, ..), d'un réseau donné et se retrouvant dans un cet emplacement considéré, n'ont pas les mêmes états, capacités et informations que d'autres composants du même réseau dans un autre emplacement géographique.

Nous définissons alors une nouvelle notion consistant à décomposer l'environnement en zones géographique que nous appelons « zones actives géographiques » (geographic active zone).

Chaque zone active géographique représentera ainsi un emplacement donné qui regroupe un ensemble d'entités de l'environnement.

Il faut bien choisir les limites de chaque zone active géographique pour mieux optimiser les échanges, en autres termes, pour mieux rapprocher les *Avatars* s'exécutant dans ces zones aux leurs entités réelles.

Ici s'introduit ainsi, la notion de mobilité physique des *Avatars*. Un *Avatar* suivra donc les déplacements de son entité qu'il représente. Si l'entité se déplace et change de zone géographique, son *Avatar* se « migrera » ainsi pour passer de la première zone active géographique à la deuxième.

Une zone active géographique permettra donc au système de regrouper plus facilement les *Avatars* des entités qui sont physiquement proches entre elles, et qui peuvent partager un certain ensemble d'information contextuelles, de services ou même des décisions. Ainsi, il est envisageable que l'*Avatar* X d'un utilisateur donné, peut bénéficier des informations contextuelles en possession d'un *Avatar* Y représentant un autre utilisateur co-localisé avec le premier. Ces informations peuvent concerner par exemple l'ensemble des réseaux disponibles et leurs charges respectives. Cette coopération entre *Avatars* d'une même zone géographique peut contribuer à minimiser les échanges d'information de contexte dans le système.

Les *Avatars*, pourront donc se déplacer dans l'architecture de Virtualisation, à la manière des agents mobiles, afin de suivre et d'être toujours au plus près de l'entité représentée.

V.2.2. Mobilité Logique des *Avatars*

Plusieurs aspects et services sont partagés par des ensembles déterminés d'*Avatars*. En effet, certains *Avatars* ont plusieurs points en commun autres que la co-localisation de leurs entités représentées. Ces points en commun peuvent concerner certains aspects tels que le type de l'entité représentée (utilisateurs grand publique, utilisateurs professionnels, …), l'utilisation du système (par exemple groupe d'utilisateurs collaboratifs pour une tâche donnée ou utilisant les mêmes services) ou bien certains aspects qu'un ensemble d'*Avatars* partagent logiquement temporairement et dynamiquement. Ainsi le système pourra regrouper ces *Avatars* dans des groupes logiques permettant alors de les séparer des autres *Avatars*. Nous introduisons ici la

notion d'un espace virtuel regroupant un ensemble d'*Avatars* que nous appelons « zone active logique ».

Une « zone active logique » regroupe logiquement un ensemble d'*Avatars* partageant des aspects et des utilisations communes. Cette séparation logique permettra d'optimiser les échanges des informations et notifications au sein du système. En effet, les messages échangés entre ces *Avatars* regroupés ne concernent pas (aspect de sécurité) les autres *Avatars* ou groupes d'*Avatars* et vice versa. D'un autre côté, la collaboration dans les prises de décision sera optimisée aussi au sein d'une même zone active logique.

Par exemple, considérons un ensemble d'utilisateurs représentant une entité (telle qu'une entreprise). Ces utilisateurs même qu'ils sont mobiles physiquement, ils restent toujours des membres collaboratifs pour atteindre les objectifs de l'entité qu'ils représentent (professionnels). Regroupant ces utilisateurs dans une zone active logique permettra de

- garder l'aspect de sécurité exigé par la nature de leurs appartenance et utilisations professionnelles ;
- mais permettra aussi d'optimiser les échanges des informations contextuelles qui seront restreints aux utilisations au sein de cette même entité.

Un *Avatar* peut sortir d'une zone active logique. En effet, l'appartenance d'un *Avatar* à une zone active logique donnée exige que cet *Avatar* partage avec les autres membres de cette zone les mêmes aspects et/ou utilisations, et si ce n'est plus le cas cet *Avatar* doit quitter cette zone. La *mobilité logique* des *Avatars* entre les zones actives logiques est donc définie.

Les deux aspects de regroupement des *Avatars* peuvent être combinés ou indépendants. En effet, nous pouvons avoir la coexistence de plusieurs zones actives physiques et plusieurs zones actives logiques dans le système. Un chevauchement d'appartenance des *Avatars* est possible. En effet, un *Avatar* donné peut appartenir à

- une zone active physique en considérant la géolocalisation physique de son entité représentée dans le système,
- une zone active logique si cet *Avatar* partage des utilisations particulières et spécifiques avec d'autres *Avatars* (qui peuvent être dans d'autres zones active physiques)

Ainsi, la mobilité logique consiste à la migration des *Avatars*

- d'une zone active logique à une autre
- d'une zone active logique à une zone active physique (s'il n'appartient plus à aucune autre zone active logique).

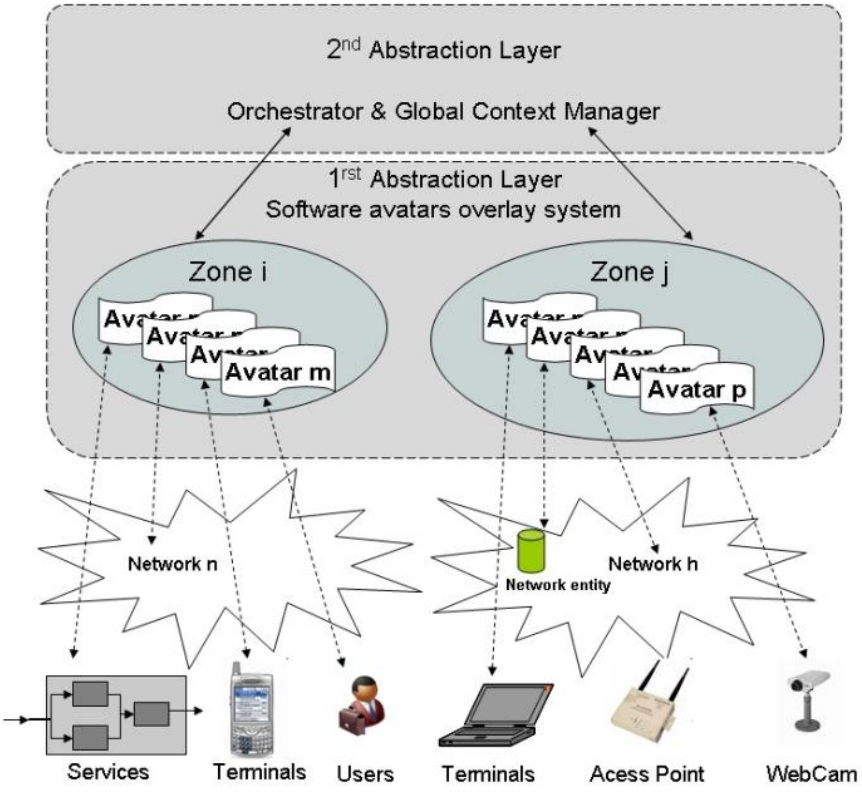


Figure 33 : Une architecture sur deux niveaux d'abstraction

VI. Un deuxième niveau d'abstraction : Une base de données sémantique pour la gestion du contexte

L'environnement considéré dans notre problématique étant ouvert, extensible et imposant des contraintes essentiellement liées à l'hétérogénéité, rend la gestion du contexte et les processus de prise de décisions des taches lourdes et complexe.

La distribution introduite par le premier niveau d'abstraction, présenté plus haut, peut engendrer des difficultés de gestion et des risques de conflits globaux. Ainsi, et comme nous l'avons introduit dans les besoins issus de notre problématique considéré, une vue globale du système devient une nécessité dans un tel environnement. Nous proposons que cette vue globale soit centralisée et séparée logiquement du niveau plus bas (proche aux entités de l'environnement, définit par les zones regroupant les *Avatars*). Nous introduisant ainsi le niveau plus haut qui devient le deuxième niveau d'abstraction de notre architecture.

Ce deuxième niveau d'abstraction aura une vue globale et sera responsable de la gestion des informations de contexte statique, de l'orchestration globale et de la gestion des interactions au sein du système.

L'utilisation d'une ontologie, décrivant l'environnement considéré, dans ce niveau plus haut permettra au système d'unifier le langage de communication entre ses composants (utilisation des concepts, attributs et propriétés, etc.). En fait, l'utilisation d'ontologies pour modéliser le contexte offre à l'infrastructure à notre architecture un modèle formel, ce qui facilite le raisonnement sur les connaissances de contexte et les rend interprétables par une machine, à savoir les composants logiciels du premier niveau, les *Avatars*.

Ainsi, pour assurer l'interopérabilité des différents dispositifs et le partage de l'information de contexte, les dispositifs et composant qui utilisent l'architecture que nous proposons utilisent une même ontologie, *Ubiquity-Ont*, pour décrire leurs informations de contexte. En effet, nous pouvons bénéficier de la sémantique introduite aux informations contextuelles avec l'ontologie pour homogénéiser les échanges entre les composants de notre architecture et ainsi faciliter l'automatisation et le raisonnement des taches et des actions de prise de décision.

Ce niveau plus haut pourra ainsi compiler les données contextuelles globales et gérer leurs historiques afin d'en tirer des règles d'optimisation basées sur l'apprentissage. Ainsi les fonctions principales de ce deuxième niveau d'abstraction de notre architecture sont présentées dans ce qui suit.

VI.1. Gestion des informations de contexte statiques

Ce niveau gère les informations de contextes statiques. Ces informations définissent et décrivent l'ensemble des intervenants et composants du système et de l'environnement.

La fréquence de mise à jour de ces informations est lente ; En effet, elle se fait soit manuellement volontairement par l'entité en question (changement de préférences ou de caractéristiques, exigences, etc.) ou bien d'une façon automatique par le système lui-même quand il détecte un changement après raisonnement.

Dans la partie consistante de notre gestionnaire de contexte, nous avons étudié l'apport de la notion d'ontologie pour améliorer l'interopérabilité et aide à l'automatisation et l'unification des interfaces de communications inter-composant hétérogènes. En effet, l'ontologie que nous avons proposée dans le chapitre précédent qui décrit un tel environnement sera utilisée. Cette ontologie définit les objets de cet environnement (utilisateurs, équipements, réseaux, services) et toutes les informations de contexte statiques qui les concernent tels que leurs profils, préférences, caractéristiques, capacités, exigences, etc.

Le stockage de ces informations permet aux systèmes pervasifs de s'adapter à chaque profil et exigences/capacités ; Ce niveau instancie à chaque intégration d'un objet au système le concept/classe concerné dans l'ontologie. Nous obtenons à ce niveau une base de données sémantique comportant deux parties :

- Base de connaissances (Tbox, c.f. chapitre 3) : toutes les données qui décrivent les concepts, leurs propriétés, les relations et les restrictions entre ces concepts liées à chaque entité de l'environnement, ces données sont enrichies par la sémantique assurée par les relations et les propriétés sémantiques entre les classes/concepts définissants ces entités dans l'ontologie.
- Base d'instances (Abox, c.f. chapitre 3) : toutes les instances des concepts de la base de connaissances, donc les valeurs de leurs propriétés et attributs.

Ces deux parties constituent la base de données sémantique (c.f. Figure 34 : Utilisation de l'ontologie pour la gestion du contexte). Ainsi, ce second niveau d'abstraction fournit et met à la disposition des autres composants de l'architecture cette base de données sémantique. Un gestionnaire de cette base est nécessaire pour répondre aux requêtes reçues pour pouvoir

extraire les données demandées. Ce composant est décrit dans le chapitre précédent (c.f. section IV du chapitre 3).

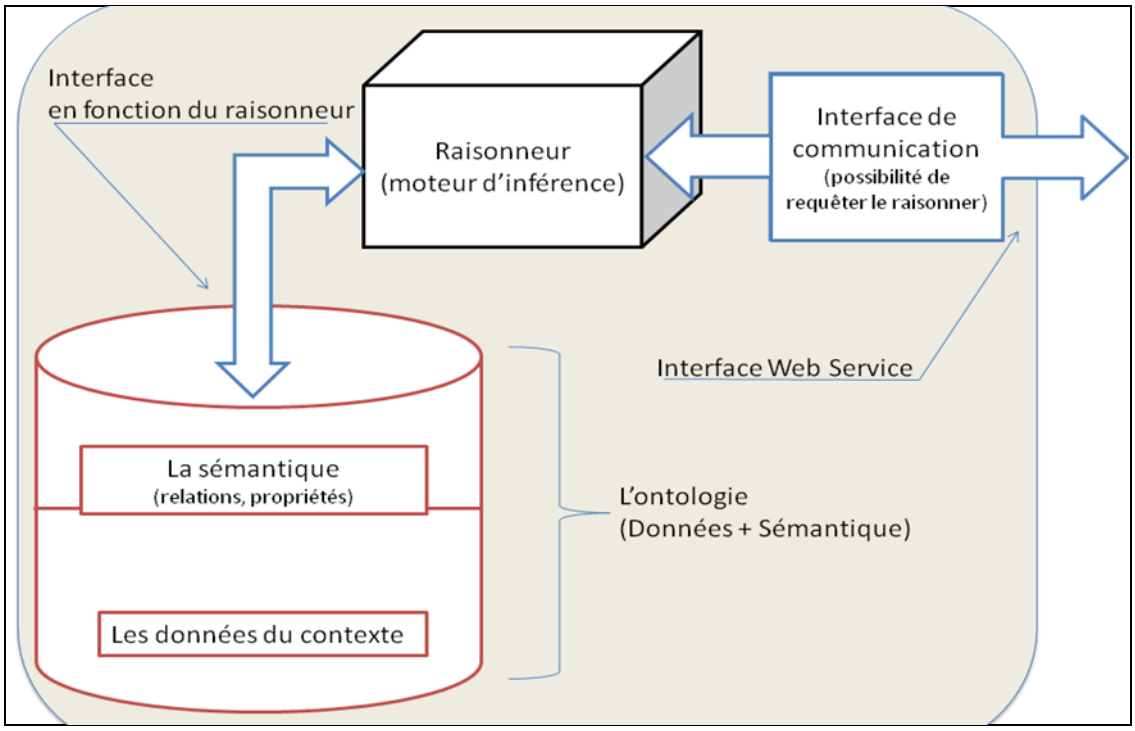


Figure 34 : Utilisation de l'ontologie pour la gestion du contexte

Ainsi, l'ontologie proposée joue le rôle d'une base de données sémantique. La sémantique qu'on peut introduire avec l'expressivité des langages des ontologies se traduit par la définition d'un ensemble de relations entre les concepts, des propriétés de ces derniers ainsi qu'un ensemble de restrictions et d'axiomes.

VI.2. Orchestration globale :

Ce second niveau d'abstraction a une vue globale sur l'ensemble des composants de celui plus bas. En effet, c'est à ce niveau que le système gère les zones active physiques initialement crées et c'est lui aussi qui gère les zones actives logique (les crée en sélectionnant les *Avatars* concernés et c'est lui qui gère la mobilité de ces *Avatars*). D'un autre côté, ce niveau plus haut a

un rôle de superviseur du fonctionnement des *Avatars* au sein de chaque zone (physique ou logique). Il pourra intervenir en proposant des modifications des décisions à prendre pour anticiper et prévenir des éventuels conflits. Ce raisonnement à ce niveau global permet d'améliorer les performances du système dans sa globalité. Un block fonctionnel exécutant ces raisonnements est alors intégré dans ce niveau d'abstraction de notre architecture, il est l'orchestrateur global du système nous l'appelons « *inference manager* » (c.f. Figure 34).

Ce niveau plus haut communique avec celui plus bas via une interface unifiée similaire à celle entre les *Avatars* et leurs entités représentées. En effet, à ce niveau plus haut, chaque zone est représentée par un *Avatar système* qui la gère. L'ensemble de ces *Avatars système* constitue l'interface de communication entre les composants du niveau plus bas et les blocks fonctionnels de ce niveau plus haut. Cet ensemble nous l'appelons « transaction process » (c.f. Figure 35).

VI.3. Gestion de l'historique et raisonnement global

L'historique des informations de contexte peut se révéler très utile pour les environnements d'informatique diffuse, notamment pour inférer le futur à partir du passé, et adapter son comportement aux habitudes et aux préférences des utilisateurs. Le flux d'informations circulant entre les producteurs d'informations de contexte et les consommateurs est caché des utilisateurs, une entité ou un composant informatique doit intercepter ces informations pour mettre à jour la base des données contextuelles.

Ce niveau plus haut, ayant une vue globale est le meilleur candidat d'incorporer et exécuter ces fonctions de ce block fonctionnel gérant l'historique des données de contexte dynamiques des entités. Ainsi, il s'occupe de la collecte et du stockage de ces informations. Ainsi nous enrichissons et adaptons l'ontologie proposée par des attributs, propriétés et relations supplémentaires à chaque classe/concepts définissant une entité/composant qu'on gère l'historique de ses informations de contexte dynamique.

Ce block fonctionnel est intégré dans l'orchestrateur global du système « *inference manager* » décrit plus haut. Cette intégration est justifiée par le fait que les informations issues par la gestion de l'historique des informations de contexte dynamique aideront à l'orchestration et l'optimisation du fonctionnement global du système.

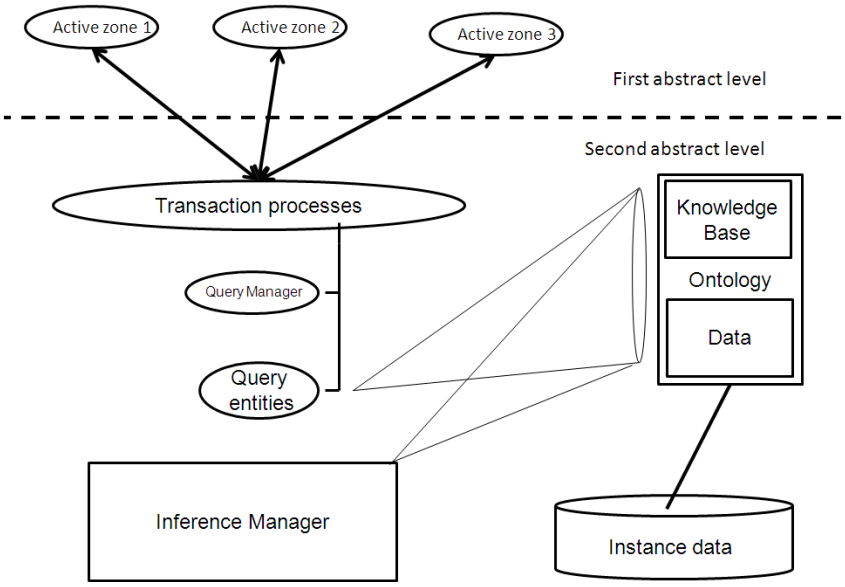


Figure 35 : Deuxième niveau d'abstraction

Conclusion

Dans ce chapitre nous avons présenté notre proposition architecturale pour un système pervasif. Cette architecture proposée, permet en effet d'ajouter à l'environnement les composants nécessaires pour à la gestion de contexte et offre les capacités de développer des services à valeur ajoutée consommant les informations de contexte. Cette architecture consiste en un overlay basé sur deux plans conceptuels de contrôle. L'idée d'avoir deux plans de contrôle est reliée essentiellement à la dynamicité des informations contextuelles dans chaque plan. Le premier plan, le plus « actif » et « dynamique » gère et utilise les informations de contexte les plus dynamiques. Par contre le deuxième plan, le plus « statique » et le moins dynamique gère les informations contextuelles qui changent peu dans le temps. Ce deuxième plan contrôle aussi le premier plan, puisqu'il a une vue plus globale sur tout le système.

Nous avons introduit d'autres aspects non fonctionnels, tels que la mobilité et la définition de zones actives (physique et logique) qui optimisent l'architecture et pour la rendre plus extensible et plus flexible. Ces zones sont définies dans le premier niveau le plus proche des entités de l'environnement. Ces zones regroupent des composants logiciels autonomes qui

représentent les entités réelles de l'environnement dans l'architecture. Nous définissons donc ici la notion des Avatars. Un Avatar est donc un composant logiciel autonome capable de gérer le contexte d'une entité réelle et de prendre à sa place les décisions adéquates pour s'adapter aux changements du contexte.

Une zone active sera définie selon le système réel et ses utilisations. Une zone active peut être celle la plus proche géographiquement aux utilisateurs, ou bien celle qui regroupe des utilisateurs d'une même catégorie ou aussi celle qui regroupe des équipements/services d'une même classe. Ainsi selon le cas d'études à mettre en place notre architecture doit être raffinée ; par exemple si le projet c'est un système domo-pervasif (le domicile d'un utilisateur) les zones peuvent être définies selon la concentration des équipements dans la maison, par exemple une zone pour le salon, une autre pour la cuisine et une autre pour le jardin. Et si le client voudra une continuité même dans son bureau alors une autre zone sera définie.

Un deuxième niveau contenant une un gestionnaire de contexte plus sophistiqué et un gestionnaire de zones actives. En effet, le gestionnaire de contexte dans ce niveau consiste à un système à base d'ontologie pour la description des éléments contextuels et un module de raisonnement et d'inférence. Ce module utilise le ou les ontologie(s) pour extraire les informations contextuelles (instances des concepts d'ontologie) suite à des requêtes. Le détail de la conception et les méthodologies adoptées de l'ontologie de notre système a été présenté dans un article publié (voir bibliographie).

Chapitre 5 : Cas d'études

I. Introduction

Ce chapitre est dédié à l'étude de deux cas d'utilisation qui vont nous permettre de vérifier nos deux propositions d'ontologie et d'architecture présentés dans les chapitres précédents.

Nous considérons que ces études font partie et complète la méthodologie proposé dans la chapitre 3 et 4 car ils permettent de vérifier nos propositions, de les affiner mais aussi d'en évaluer leurs performances fonctionnelles et quantitatives

Les deux études considérées concernant des problématiques d'actualité dans le domaine des réseaux et des services de télécommunication, à savoir :

- La gestion de la mobilité et la continuité de services sur un environnement de réseaux d'accès sans fil hétérogènes.
- L'optimisation de la consommation d'énergie sur les terminaux mobiles sur ce même environnement.

Les propositions de cette thèse vont donc être confrontées à des problématiques réelles issues de projets de recherche réalisés au sein de l'équipe d'accueil.

- Le premier, le projet ANR/SEAMLESS, vise à proposer des solutions et des mécanismes coopératifs innovants, permettant d'assurer une coopération basée sur la virtualisation et la gestion de contexte. Le projet considère un ensemble de technologies de réseaux d'accès et un ensemble de services professionnel et grand public. Le but, étant de garantir la continuité de service tout en permettant aux utilisateurs de se connecter au réseau le plus approprié. Le projet est axé sur deux aspects à optimiser : la mobilité verticale (handover entre plusieurs technologies) et l'adaptation ou la recomposition des services en fonction du contexte de l'utilisateur.
- Le second, le projet FUI/Wonderville, concerne d'autres aspects aussi importants. Il vise à proposer des solutions pratiques pour des villes connectée et écologiques. Outre le but d'offrir des sévices centrés utilisateurs au sein de la ville, un des objectifs

principaux et de garantir un niveau minimum de la consommation énergétique des technologies et des solutions utilisées. Ces aspects font partie de la problématique plus générale des télécommunications vertes (Green Networking).

Dans ce qui suit, nous allons confronter nos propositions à ces deux cas d'études et analyser leurs performances. Les améliorations nécessaires sur l'architecture et l'ontologie proposées dans cette thèse seront proposées et explicitées.

II. Premier cas d'étude : Projet SEAMLES pour l'optimisation de la mobilité inter-réseaux et l'adaptation de services

II.1. Problématique, Objectifs et contraintes

La gestion de la mobilité sur des environnements de réseaux d'accès sans fil et hétérogènes a été largement abordée dans la littérature. Les principales difficultés liées à cette problématique sont liées au manque de partage d'information et de coopération entre les systèmes concernés.

Le cadre précis de ce premier cas d'utilisation est résumé sur la Figure 36.

Il s'agit de l'environnement des agents et des passagers de la RATP (Régie Autonome des Transports Parisiens). Cet environnement est riche en technologies réseau et en services.

Nous considérons ici des utilisateurs avec des terminaux multimodaux, pouvant bénéficier de l'accès à plusieurs technologies d'accès radio tel que le WiFi, la 2G et la 3G et de différents services professionnel ou particuliers.

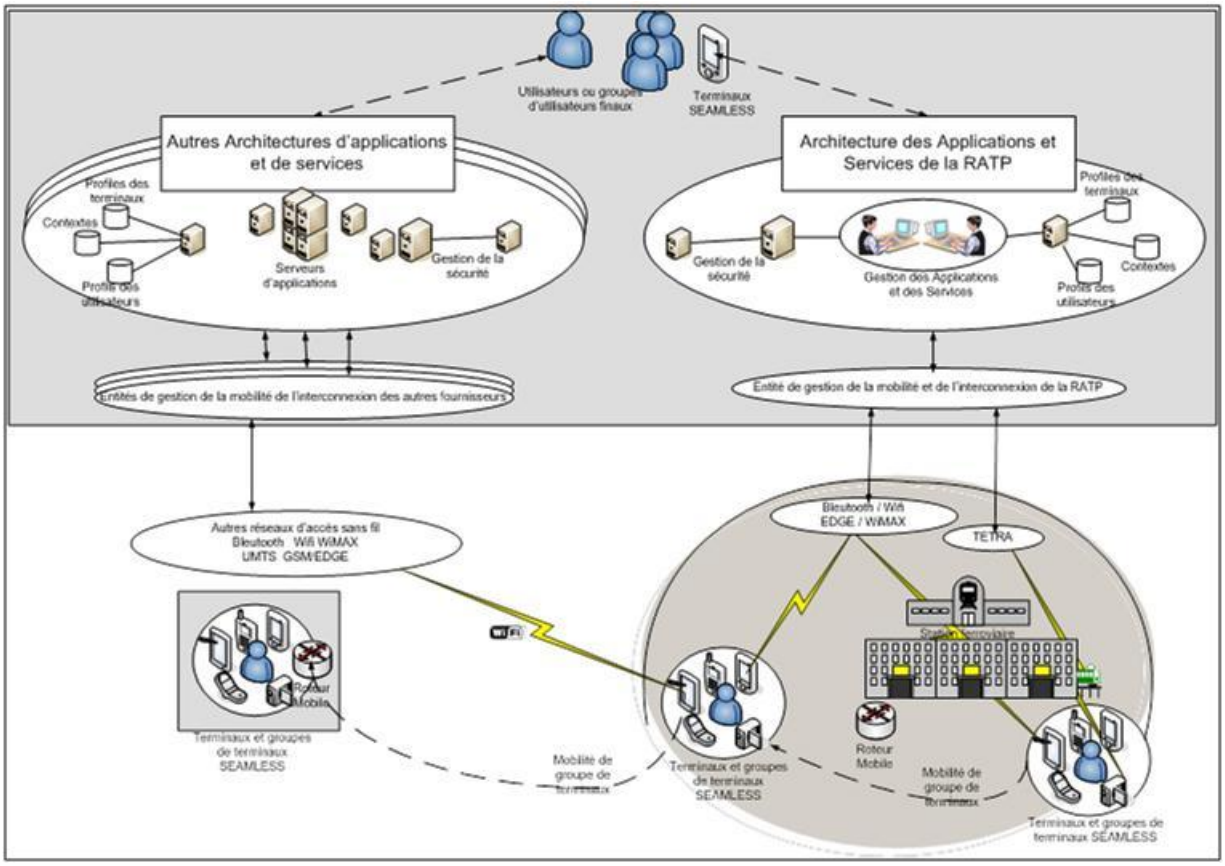


Figure 36 : Environnement considéré pour le premier cas d'étude

L'environnement est composé des éléments et concepts suivants :

- utilisateurs (agents ou passagers) possédant un ou plusieurs terminaux,
- terminaux avec des caractéristiques et des capacités différentes, dont certains sont multimodaux,
- réseaux et équipements réseaux avec différentes technologies : 2G, 3G, WiFi,
- des applications et services.

Il s'agit alors de faciliter et de rendre transparent aux utilisateurs le handover vertical afin de garantir la continuité des services. Cette opération consiste à permettre aux terminaux mobiles de passer d'une technologie d'accès à une autre, en fonction du contexte réseau, sans expérimenter des coupures. Cette problématique fait appel à la notion de handover vertical, en opposition au handover horizontal effectué sur une même technologie.

Le handover horizontal est un mécanisme réseau qui permet à un terminal mobile de changer son point d'attachement (Station de Base) au sein d'un même réseau. Cette opération, a été largement étudiée dans les réseaux cellulaires de type 2G et 3G. Elle bénéficie de mécanismes ayant déjà fait leurs preuves. Le choix est principalement effectué sur la base des niveaux de signal reçu par le mobile et de la charge des cellules qui le couvre. Sur les technologies cellulaires tel que le GSM ou l'UMTS, les standards ont prévu les opérations permettant au mobile de scruter son entourage pour détecter les cellules avoisinantes. Par conséquent, les informations nécessaires à effectuer le bon choix sont disponibles soit sur le terminal, soit sur le réseau. Les algorithmes de décision pour le handover horizontal sont assez murs et très performants. Ils permettent aux réseaux cellulaires d'offrir de la mobilité intra-système, sans coupure et permettant une bonne continuité de service au sein de la même technologie.

Mais dès qu'on considère le handover vertical, sujet de la présente problématique, on se retrouve confronté à plusieurs difficultés.

Le handover vertical est la procédure permettant à un mobile de changer son point d'attachement (Station de Base ou Point d'accès) pouvant appartenir à un autre opérateur que celui auquel il était préalablement connecté ou utilisant une autre technologie. Un exemple est le passage d'un réseau 3G à un réseau WiFi.

Le Handover Vertical pose plusieurs défis et problématiques :

- Le manque d'information sur les réseaux disponibles en termes de qualité de signal, de QoS, de charge et de coût. En effet, le terminal, généralement connecté à un seul réseau, n'a aucun moyen de récupérer des informations sur les autres réseaux, sans s'y connecter. Même s'il existe des propositions permettant à un réseau de diffuser à ses abonnées des informations sur les autres réseaux voisins (norme 802.21), il n'existe jusqu'ici ni d'architecture ni d'implémentation réelle.
- Même en supposant que toutes les informations nécessaires sur les différents réseaux dans le voisinage du terminal sont disponibles, les algorithmes de prise de décision deviennent complexes vu le nombre de paramètres à considérer et ne peuvent être délégués au terminal mobile, qui peut ne pas posséder assez de ressources.
- La prise de décision par un des réseaux disponible peut être subjective, favorisant se même réseau, et probablement impossible en l'absence de mécanismes et d'accords de coopération entre les différents réseaux.

Un autre aspect lié à cette problématique de mobilité verticale, concerne l'adaptation des services offerts. En effet, vu que les différents réseaux, utilisant différentes technologies, n'offrent pas les mêmes niveaux de QoS en termes de débits et de garanties sur les délais, les services en cour d'exécution doivent être adaptés ou recomposés à la volé après la phase de handover vertical. Les décisions d'adaptation et de recomposition auront également besoin de connaissance provenant des terminaux mobiles et des réseaux considérés, mais également de la plateforme de service en supposant que celle-ci offre les composants de services nécessaire à l'adaptation.

Nous pouvons donc englober dans la notion de service, ceux qui concernent les aspects réseaux, de bas niveau tel que la gestion du handover et de la mobilité, et ceux de haut niveau liés aux applications informatiques utilisées par les abonnés tel que le streaming vidéo et l'envoi de fichiers.

Ces problématiques rentrent dans le cadre des systèmes dits auto-adaptatifs, capables de s'autogérer et de prendre automatiquement des décisions optimales en fonction de chaque contexte.

Pour apporter des solutions à ces problématiques, le premier aspect à considérer est la disponibilité des informations nécessaires pour les prises de décision. Or sur un système aussi hétérogène, la disponibilité de ces information nécessite des mécanismes d'harmonisation et de coopération inter systèmes et inter couches. Il s'agit en effet de permettre l'échange

- des différentes mesures réseaux provenant des couches basses sur les différentes technologies (signal, débit, localisation des terminaux, …)
- des informations provenant des couches intermédiaires concernant l'état des réseaux (routages, QoS, …)
- des informations provenant des couches hautes liées à la disponibilité des services et des composants nécessaires à leur adaptation ou recomposition

On peut également considérer les informations liées à la sécurité et au coût (que nous n'allons pas traiter dans ce cas d'étude).

La richesse et la complexité d'un tel environnement nécessitent des mécanismes plutôt automatisés, capables de collecter et de raisonner sur des informations de contexte provenant de sources hétérogènes.

- Collecte des informations de contexte sur l'environnement hétérogène,
- Analyse de ce contexte,
- Prise des décisions les plus adaptées,
- Notification pour l'exécution de ces décisions,

Il est également évident pour des raisons de performances et de limitation sur les terminaux mobiles, de décharger ces derniers de ces fonctions.

D'un autre côté, deux autres aspects nous semblent importants à souligner. En effet, pour pouvoir prendre des décisions tenant compte de tout l'environnement avec toute sa complexité, il faudra :

- Avoir une vue globale sur l'ensemble de l'environnement
- Penser à minimiser les échanges nécessaires et à réduire la complexité des algorithmes de décision, en introduisant par exemple des mécanismes d'apprentissage.

Nous pensons que les résultats obtenus dans les chapitres précédents, et en particulier l'architecture de virtualisation et l'ontologie que nous avons proposées, peuvent apporter une solution originale à ces problématiques.

II.2. Analyse détaillée « Bottom-Up »

Nous commençons cette analyse par la définition de deux scénarios d'utilisation du système. A travers ces scénarios nous allons détailler le cahier des charges incluant les blocks fonctionnels nécessaires.

Les deux scénarios ont été choisis parmi un panel d'autres scénarios car ils présentent, à notre sens, les exigences les plus fortes en termes techniques.

II.2.1. Scénario 1: Usage professionnel

Ce premier scénario concerne une intervention technique d'un agent de la RATP pour identifier et résoudre une panne. L'utilisateur, un agent technique, a reçu sur son terminal mobile (tablette ou smart phone) une alerte sur un incident. Grâce à l'interface dédiée, il a pu récupérer certaines

informations relatives à cet incident et visionner en direct le lieu de l'incident en accédant à une caméra de vidéosurveillance sur le lieu de l'incident. Il s'est ensuite rendu sur place, sans perdre sa connexion grâce au réseau WiFi disponible dans le bus de la RATP. Un routeur mobile est en effet embarqué dans le bus et permet d'offrir du WiFi tout en utilisant des interfaces WiFi/3G pour se connecter à Internet. Une fois sur place, l'utilisateur a pu résoudre la panne et clôturer cet incident tout en informant son supérieur hiérarchique par visioconférence.

II.2.2. Scénario 2: Usage grand public

Ce scénario concerne un habitant lambda de la ville. Cette personne dispose d'un Smartphone ayant les connectivités 3G et WiFi ainsi que des capacités logicielles et matérielles pour bénéficier du service de visioconférence. Le scénario commence à la maison avec l'initiation du service de visioconférence. Après un certain temps, la personne quitte son domicile pour faire ses courses tout en gardant la session de la visioconférence active. La session de visioconférence doit donc être basculée en douceur de l'accès WiFi disponible à la maison sur un accès 3G disponible dans la rue. Ensuite, et en prenant le bus, le Smartphone se connecte au meilleur réseau disponible qui est le WiFi desservit par le routeur mobile installé dans le bus et offrant un meilleur débit que la 3G.

II.2.3. Identification des acteurs et entités du système

A travers l'analyse de ces deux scénarios, nous pouvons dans une première étape identifier les principales entités et acteurs du système.

- Environnement : ville, domicile, bureau, réseaux de transport (train, métro, bus, stations)
- Utilisateurs :
    - Particuliers, habitant la ville et pouvant accéder à des services grand public
    - Agents professionnels pouvant accéder à des services professionnels
- Terminaux : Smartphone, tablettes, PC de bureau ou portables
- Réseaux : WiFi, 2G, 3G ayant différentes technologies et différentes couvertures.

- Services :
    - Services grand public : accès Web, Visioconférence, …
    - Services professionnels : alertes, accès à une caméra de surveillance

II.2.4. Spécification de l'architecture et des entités fonctionnelles

A partir de la précédente analyse des scénarios, et l'identification des principaux acteurs et entités du système, nous pouvons maintenant définir les blocks fonctionnels nécessaires et leurs interactions au sein du système SEAMLESS.

Les scénarios d'usage font ressortir deux fonctionnalités centrales :

- Mobilité inter-réseaux sans coupure (seamless vertical handover)
- Adaptation des services en fonction des capacités des terminaux de l'utilisateur et des ressources réseaux disponibles

Ces deux fonctionnalités nécessitent l'accès et le raisonnement sur les informations de contexte. Un ensemble d'outils est donc nécessaire pour réaliser ces fonctionnalités.

Commençons par énumérer les entités fonctionnelles qui répondent aux besoins et aux spécifications discutés plus haut :

- Des entités fonctionnelles Services pour le grand public
- Des entités fonctionnelles Services pour les agents professionnels
- Une entité fonctionnelle pour faciliter l'adaptation des services
- Une entité fonctionnelle pour la gestion du contexte
- Un gestionnaire de mobilité inter-réseaux

Nous proposons également un block fonctionnel pour orchestrer tous ces aspects et veiller à ce que les différents composants fonctionnent en harmonie :

- Le block Orchestrateur du système.

Ces différentes entités fonctionnelles et leurs interactions sont présentées sur la Figure 37.

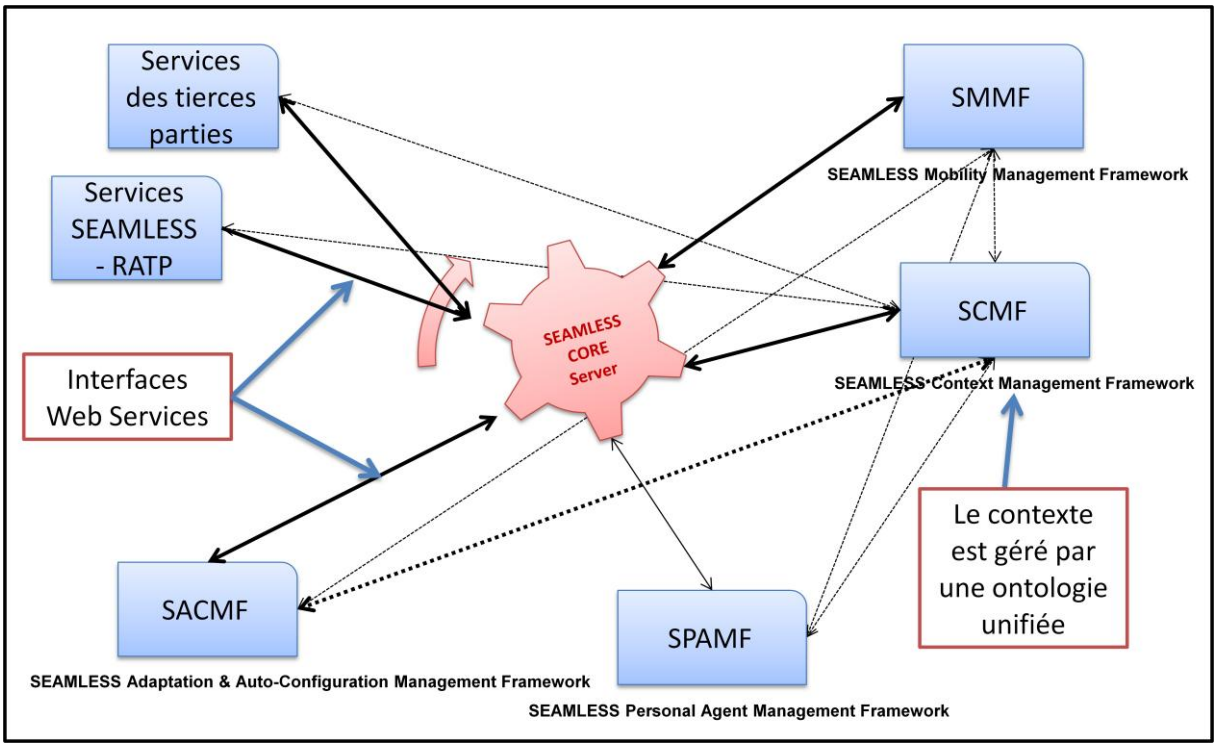


Figure 37 : Les principales entités fonctionnelles du système SEAMLESS

II.2.4.1. L'entité SCMF (SEAMLESS Context Management Framework)

La gestion de contexte est une fonctionnalité primordiale à tout le système SEAMLESS. C'est sur sa base que les opérations de gestion de mobilité et d'adaptation de service vont se dérouler.

L'entité fonctionnelle SCMF (SEAMLESS Context Management Framework) doit être capable de récupérer et de mettre à disposition des autres entités, les différentes informations

Elle doit prendre en charge l'acquisition (ou la collecte), l'interprétation, la représentation et le stockage des données contextuelles, puis de les mettre à disposition des autres entités consommatrices de contexte. Etant l'élément central de l'architecture proposée, cette entité sera détaillée dans les sections suivantes.

II.2.4.2. L'entité SMMF (SEAMLESS Mobility Management Framework)

Cette entité est responsable de la gestion de la mobilité inter-réseaux. En se basant sur les informations de contexte, elle déroule des algorithmes de prise de décision pour le handover Vertical.

L'entité SMMF (SEAMLESS Mobility Management Framework) est composée de plusieurs sous-entités fonctionnelles (c.f. Figure 38 : Aperçu sur le SMMF):

- **L'entité Gestion des Handover** : cette entité a pour rôle de déclencher les opérations de Handover vertical. Elle est responsable de la prise des décisions et de l'initiation de l'exécution (allocation des ressources dans le nouveau réseau d'accueil et libération des ressources sur le réseau à quitter).

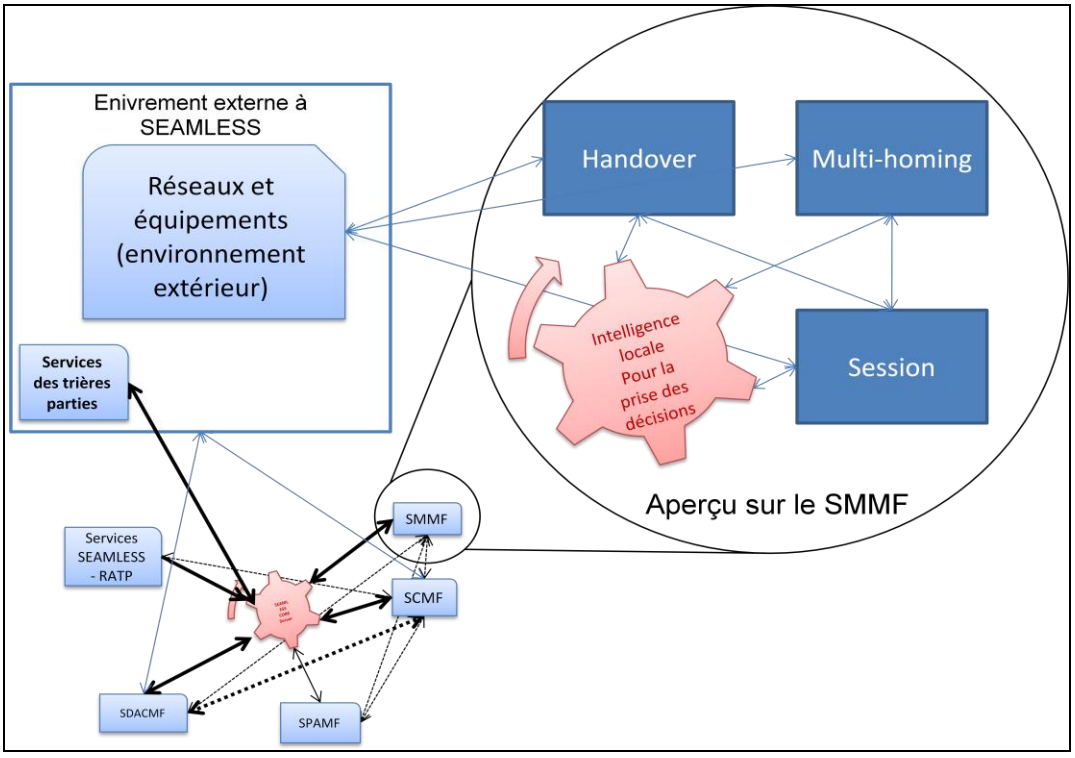


Figure 38 : Aperçu sur le SMMF

- **L'entité Gestion du Multi-Homing** : cette entité est responsable des décisions concernant le Multi-homing, c'est à dire l'usage simultané de plusieurs interfaces réseaux sur un même terminal. Son premier objectif est de démultiplier le débit offert à un terminal donné, en fonction des besoins des applications en cours et de la capacité offerte par les réseaux disponibles. Un deuxième cas d'usage pour cette entité concerne l'optimisation du mécanisme de handover vertical en introduisant la notion de make-before-brake. Dans ce cas, le terminal peut utiliser deux interfaces réseaux en parallèle pendant sa migration d'un réseau à un autre. Ceci permettra d'éviter les mini-coupures et les délais de transfert de sa session. Cette entité agit au niveau de la couche transport et permet de gérer des protocoles spécifiques au multi-homing tels que mSCTP et MTCP.
- **L'entité Gestion des Sessions** : cette entité est responsable de la gestion des sessions de service sur les terminaux mobiles. Elle surveille en permanence l'état de la session utilisateur et des différents services la composant.
- **L'entité Noyau**: cette entité a pour rôle de coordonner les actions des différentes autres entités composant la SMMF. Elle permet de s'assurer que les décisions prises par les différentes autres sous-entités ne génèrent pas de contradiction ou de conflits.

II.2.4.3. L'entité SACMF (SEAMLESS service Adaptation and equipment Configuration Management Framework)

Cette entité fonctionnelle est responsable de la prise des décisions liées aux adaptations de services. Ses décisions sont également basées sur les informations de contexte qu'elle peut récupérer à partir de l'entité SCMF.

Nous considérons ici que l'adaptation de service peut prendre deux formes :

- Soit à lancer une nouvelle instance du même service avec une classe de service différente,
- Soit recomposer le même service en modifiant un de ses composants (e.g. encodeur vidéo) pour l'adapter aux changements du contexte.

II.2.4.4. L'entité fonctionnelle « Orchestrateur Central » :

Vu la complexité des taches considérées et des interactions entre elles, il existe un risque que les décisions prises par ces dernières soient contradictoires ou conflictuelles. Nous proposons donc une entité fonctionnelle d'orchestration appelée « noyau du système SEAMLESS ». Elle est responsable de la coordination des différentes autres entités. Un exemple des responsabilités de ce noyau est de privilégier un handover vertical à une adaptation de service, si l'adaptation de service pourrait dégrader plus la QoS pour l'utilisateur que le handover.

Ce noyau aura donc une vue globale sur tous les autres blocks du système.

II.3. L'Architecture SEAMLESS

L'analyse précédente nous donne une vue globale des blocs fonctionnels nécessaires au système.

Il ressort également de cette analyse qu'il est nécessaire d'avoir

- un niveau conceptuel de haut niveau qui va orchestrer les différentes entités fonctionnelles
- des mécanismes pour la collecte et la gestion des données de contexte
- des mécanismes permettant le partage des informations de contexte entre différentes entités du système et ce pour niveaux réseau et services

L'architecture présentée dans le Chapitre 4 peut parfaitement répondre à ces besoins. En effet, l'architecture proposée est centrée sur la gestion du contexte et propose des outils d'harmonisation et de virtualisation, permettant d'augmenter des entités existantes par les fonctions nécessaires à la gestion et à la manipulation du contexte.

La notion d'Avatar, introduite dans le Chapitre 4, offre la notion de conteneur logique qui peut être instancié pour représenter n'importe quelle entité logique ou physique. Un Avatar peut également intégrer un ou plusieurs processus, sous forme de machine à états, pour se procurer des capacités de raisonnement.

Nous pouvons donc naturellement étendre l'architecture basée sur les deux niveaux de virtualisation et la notion d'*Avatars* pour tenir compte des besoins de ce premier cas d'étude.

Rappelons que l'architecture de l'*Avatar* est décrite dans le chapitre IV et illustrée par la Figure 29.

L'extension de l'architecture concernera essentiellement des blocs décisionnels pour la gestion de la mobilité et pour l'adaptation des services.

Ainsi, il faudra :

- Intégrer le ou les algorithmes de décision pour la gestion de la mobilité au sein du le bloc « Decision & Notification » de l'*Avatar* correspondant.
- Récupérer les informations de contexte nécessaires à ces algorithmes via le bloc « *context gathering* & updates » de l'*Avatar*.
- Gérer les informations de contexte relatives à la mobilité et aux capacités des services sous forme sémantique et les stocker au deuxième niveau de l'architecture (la base de données sémantique).

D'un autre côté, il faudra également prévoir des *Avatars* pour les entités « réseau » et « service ». Ces deux nouveaux *Avatars* viendront donc s'ajouter, dans le niveau de virtualisation proposé par l'architecture, aux *Avatars* déjà prévus pour les utilisateurs et leurs équipements.

L'entité fonctionnelle « Orchestrateur Seamless» quant à elle rejoindra le niveau haut de l'architecture qui possède la vision globale pour l'orchestration de l'ensemble des entités.

Elle aura donc accès à la base de connaissance basée sur l'ontologie et qui stocke les données globales de contexte. Elle pourra également agir sur les différents *Avatars* qui gèrent la mobilité ou l'adaptation de service afin de prévoir et d'éviter les conflits éventuels.

Il est également nécessaire d'enrichir l'Ontologie avec les informations de contexte relatives aux nouvelles entités introduites dans ce cas d'étude.

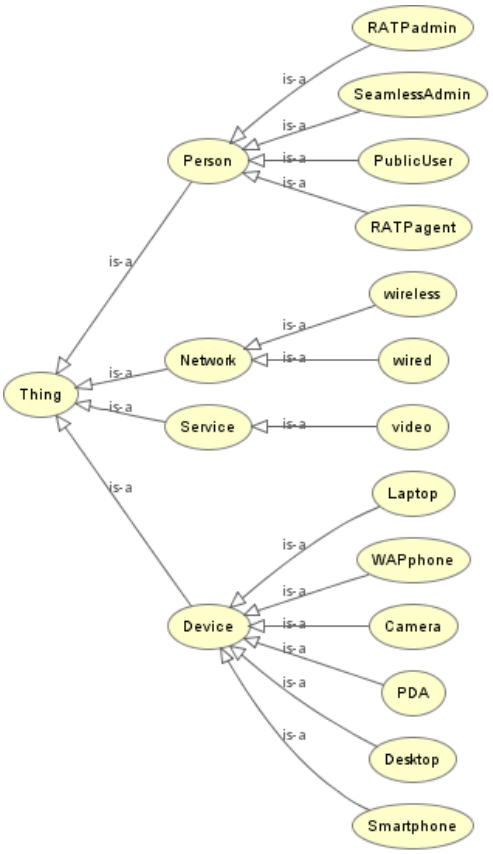


Figure 39 : Adapter l'ontologie au cas d'étude

Les informations de contexte collectées à partir de l'équipement utilisateur sont :

- Le réseau auquel le terminal est connecté actuellement ainsi que les mesures qu'il peut avoir tels que : Le taux d'erreur binaire : BER (Bit Error Rate), la gigue (jitter), le délai (delay), la bande passante,
- La liste des autres réseaux disponibles (vu par le terminal)
- Le service en cours d'utilisation ou celui invoqué
- La localisation de l'utilisateur

L'*Avatar* représentant l'utilisateur, récupèrera ces informations à l'aide de son block fonctionnel « *context gathering* » et pourra les compléter avec d'autres informations de contexte en requêtant la base de données sémantique du deuxième niveau d'abstraction. Ces informations concernent principalement :

- l'utilisateur et ses préférences, habitudes et abonnements (contexte statique);
- le ou les services disponibles, avec leurs différentes classes de Service et leurs capacités d'adaptation
- les réseaux disponibles à cette localisation, y compris celui sur lequel l'équipement utilisateur est connecté. L'*Avatar* utilisateur pourra ainsi solliciter les *Avatars* correspondants à ces réseaux pour collecter plus d'informations contextuelles, statiques et dynamiques, tel que la charge de chaque réseau et son coût.

Elles seront donc ajoutées à l'Ontologie proposée dans le chapitre III.

II.4. Introduction d'un nouveau concept : la notion de « réputation »

Dans le cadre du projet SEAMLESS, l'équipe projet a introduit une nouvelle notion basée sur la QoE (Quality of Experience), permettant à une communauté d'utilisateurs de partager leurs expériences sur un réseau donné.

La réputation d'un réseau est définie de façon dynamique et pour chaque service. C'est une mesure subjective qui prend en compte différents paramètres objectifs de QoS. Le partage de la réputation entre un groupe d'utilisateurs peut aider ces derniers à prendre des décisions plus rapides concernant leurs handovers, sans avoir besoin de consulter tous les paramètres et mesures nécessaires par les algorithmes de décision traditionnels. L'algorithme de handover vertical proposé par le projet SEALESS est décrit dans [77], [78].

Comment cette nouvelle notion peut être introduite dans l'architecture proposée ?

Pour se faire, il faut introduire deux nouvelles fonctionnalités dans l'architecture (c.f. Figure 40) :

- Mesure de la QoE, sa compilation et son partage
- Algorithme de décision pour la handover

Voici comment nous proposons d'introduire ces différentes fonctionnalités dans notre architecture (c.f. Figure 40 et Figure 41).

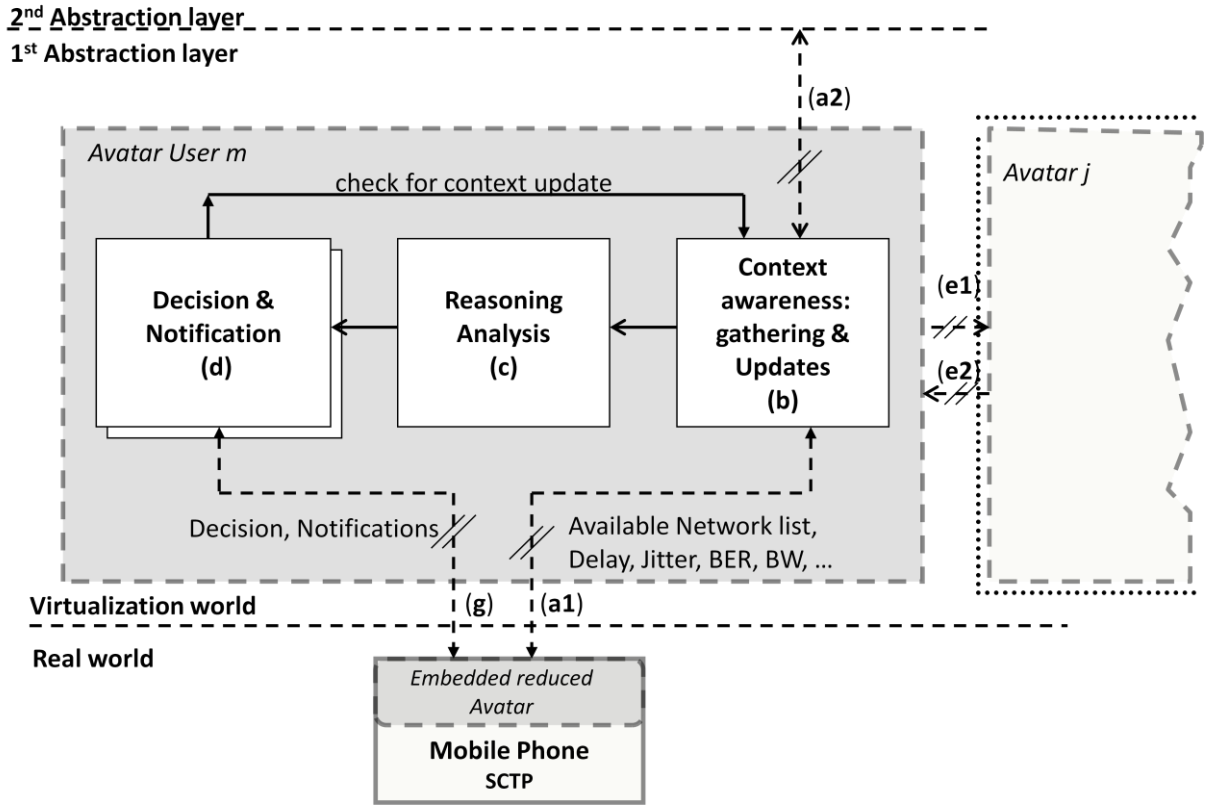


Figure 40 : Avatar utilisateur dans le cas d'étude 1.

- Modifications sur l'*Avatar* représentant l'entité réseau [79]
    - ***Bloc fonctionnel g*** : ce bloc fonctionnel permet de collecter les mesures de QoE des différents utilisateurs (via leurs *Avatars*).
    - ***Bloc fonctionnel h*** : ce bloc est responsable de la compilation des différentes QoE des utilisateurs pour chaque service. Il génère une valeur moyenne.
    - ***Bloc fonctionnel i*** : ce bloc est responsable de mettre sa valeur de réputation à disposition des différents *Avatars* qui souhaitent la consulter.

- Modifications sur l'*Avatar* représentant l'Utilisateur
    - ***Modification du bloc fonctionnel c*** : ce bloc déjà existant, est peuplé d'une nouvelle règle lui permettant de choisir parmi les différents blocs fonctionnels d.

- ***Bloc fonctionnel « d »*** : ce bloc est dédié à la gestion de la QoE et aux décisions de handover basées sur la notion de réputation. Il vient se superposer avec d'autres blocs d, responsables aussi de prendre des décisions sur le handover. Il compile les données de QoS pour en extraire une valeur de réputation du réseau auquel il est connecté.

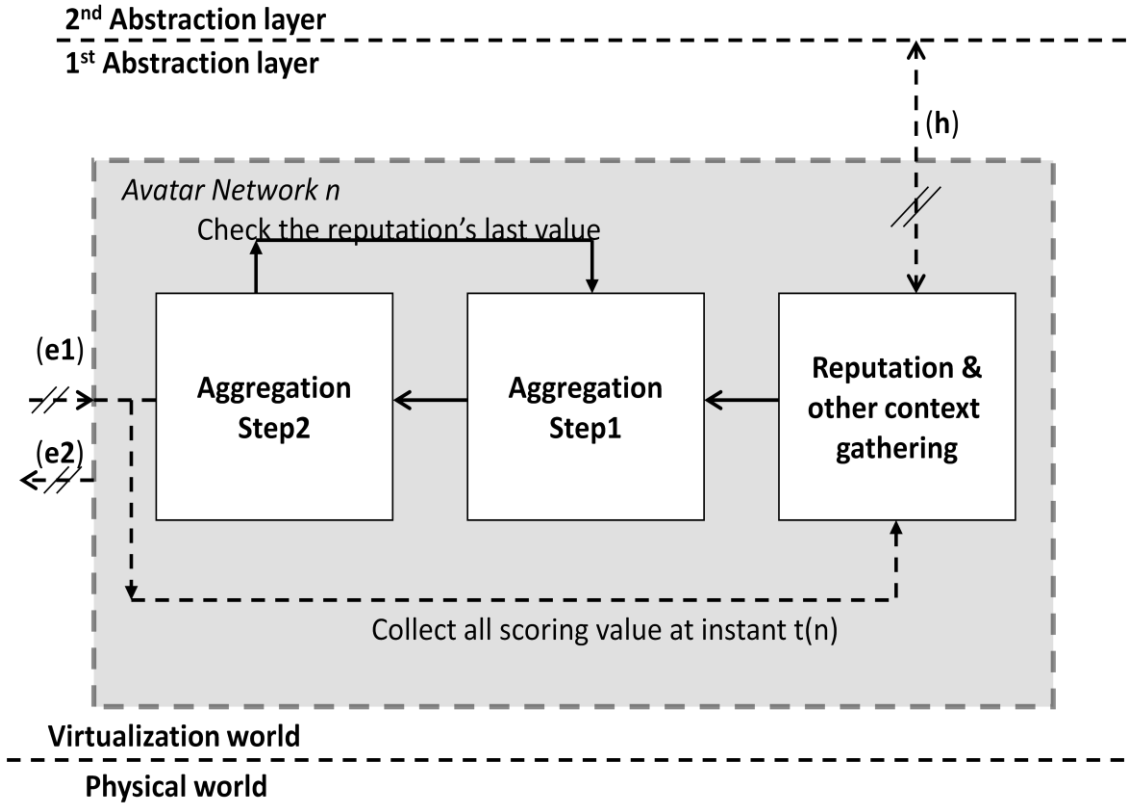


Figure 41 : Avatar d'un réseau dans le cas d'étude 1.

- Modifications sur les ***Interface e1 et e2*** : Ces interfaces entre les *Avatars* représentants les utilisateurs et les réseaux vont permettre d'échanger les valeurs des QoE et de réputation. Mais aussi toutes les informations de contexte dynamique concernant l'utilisateur, son terminal ou les réseaux en fonction des besoins.

III. Cas d'étude 2 : Projet FUI/Wonderville - Gestion des interfaces réseaux pour économiser la consommation d'énergie, ECO-WISM

III.1. Problématique

L'objectif de cette application est d'optimiser la consommation d'énergie sur des terminaux mobiles, capables de se connecter à plusieurs réseaux sans fil de différentes technologies.

La multiplication des technologies d'accès radio offre aujourd'hui une multitude de solutions réseaux, offrant chacune des caractéristiques et des services différents. Le WiFi par exemple, est de plus en plus disponible autour de nous, à la maison, sur le lieu de notre travail et aussi dans les cafés, les restaurants, les aéroports et plusieurs autres lieux d'accueil et administrations. Il offre un débit de plus en plus élevé et non encore égalé par les technologies cellulaires. Cependant, la technologie WiFi ne garantit encore ni la gestion de la mobilité ni la Qualité de Service. Elle n'est donc pas appropriée à toutes les applications. A l'opposé, nous pouvons parler des réseaux mobiles, dits aussi cellulaires, qui offrent plus de garanties en termes de QoS et de mobilité, mais pas encore assez de débit, en attendant la 4G.

Outre leurs différences en termes de performances et de services offerts, les différentes technologies d'accès possèdent des modèles de consommation d'énergie différents. Ces différences sont intrinsèquement liées aux technologies utilisées aussi au niveau physique (PHY) qu'au niveau des couches supérieures, essentiellement au niveau de la couche MAC où les techniques d'accès sont différentes et variées.

De ces différences technologiques, il résulte des différences au niveau de la consommation d'énergie. Or, pour un terminal mobile qui se base sur une batterie de faible capacité, cet aspect est primordial puisqu'il affecte directement la disponibilité des services pour l'utilisateur. Il est donc important de faire en sorte qu'une charge de batterie, dure un maximum de temps.

Comment choisir son réseau d'accès ? Quel sont les différents critères et les différentes contraintes qui doivent être considérés ?

Il est évident que l'utilisateur lambda, dans la plus part des cas, ne peut décider seul du choix du réseau auquel il se connecte, ni de la meilleure technologie à utiliser pour un besoin donné. Vue

le nombre de paramètres et de situations, une solution automatisée ou semi automatisée permettrait de l'aider à faire ce choix.

Dans la suite, nous proposons dans ce chapitre une solution architecturale basée sur la gestion du contexte et permettant aux terminaux mobiles de mieux gérer l'énergie qu'ils possèdent tout en garantissant une QoS adaptée aux services en cours.

Dans la suite de ce chapitre, nous considérons le scénario suivant :

- Soit M réseaux Ri ($1 \leq i \leq M$) d'accès sans fil
- Soit N terminaux Tj ($1 \leq j \leq N$)) mobiles capables d'accéder aux différents réseaux Mk ($1 \leq k \leq M$)
- Soit P applications Ah ($1 \leq h \leq M$) disponibles
- Chaque terminal Tj, se déplace et peut être à chaque instant dans la zone de couverture d'un ou de plusieurs réseaux.
- Chaque terminal Tj peut lancer une ou plusieurs applications à la fois
- Chaque réseau Ri utilise une technologie radio différente avec un modèle de consommation d'énergie spécifique
- Chaque application Ah génère un flux de données UP-LINK et un autre DOWN-LINK avec des contraintes et des caractéristiques différentes en termes de
    - Taille des paquets
    - Quantité de données et nombre de paquets
    - Contraintes de QoS en termes de taux de perte, de délai et de gigue

III.2. Approche

La problématique considérée nécessite un système de décision assez réactif et capable de prendre en compte plusieurs paramètres et caractéristiques du contexte provenant des réseaux, des terminaux et des applications. Une solution basée sur des informations locales ne peut pas produire une solution optimisée à l'échelle globale. Nous avons donc proposé d'étendre la solution de gestion de contexte proposée dans le chapitre quatre en l'enrichissant par des connaissance sur les modèles d'énergie de chaque technologie et de nouvelles règles d'optimisation permettant de maximiser la durée de vie de la batterie des terminaux.

III.3. Idée et Architecture proposées

Les règles utilisées pour la sélection d'interfaces pour les terminaux mobiles d'aujourd'hui sont principalement basées sur la qualité de service et la charge du réseau. Les opérateurs de réseaux cherchent principalement à réduire la charge sur leurs réseaux cellulaires en redirigeant la plupart des services de données sur les réseaux Wi-Fi si disponibles.

De plus, ces règles ne tiennent pas compte des aspects de la consommation d'énergie sur le terminal. De nouvelles règles peuvent être définies pour la sélection de l'interface afin d'optimiser à la fois la qualité de service et la consommation d'énergie.

Nous proposons ici une nouvelle approche basée sur des modèles d'énergie et sur des règles de décision tout en considérant des exigences en termes de qualité de service pour chaque flux de données.

III.3.1. Modèle d'énergie et règles

L'analyse des différentes propositions existante dans la littérature en ce qui concerne la consommation d'énergie des interfaces réseau sans fil, montre que les modèles énergétiques existants ne couvrent pas la plupart des technologies et des situations.

Nous avons compilé un certain ensemble des modèles existant afin de proposer un modèle simplifié de consommation d'énergie (voir Tableau 2 : Modèle pratique déduit d'une expérimentation) pour les trois principales technologies qui sont WiFi, EDGE et 3G.

Dans ce tableau, la consommation d'énergie est mesurée en joules. Elle est exprimée en fonction de la taille des données KB.

Data size *x* (KB)	$x \epsilon$ 1 – 4	$x \epsilon$ 4 – 10	$x \epsilon$10–50	$x \epsilon$50–100	$x\epsilon$100 – 500	$x\epsilon$500 – 1000	$x \epsilon$ 1000 – 5000
3G	0.166x+12.33	0.05 x+12.8	0.032 x+12.87	0.01 x +14	0.0125x+13.7	0.036x + 2	0.005x + 30
EDGE	0.066x + 3.73	0.083x + 3.66	0.037x + 4.12	0.06x + 3.0	0.027x + 6.25	-	-
Wifi	0.033x + 1.86	0.083x + 2.16	0.01x + 2.4	0.002x + 2.8	0.005x + 2.5	0.004x + 8	0.005x + 6.25
Wifi+SA	0.033x + 6.96	0.083x + 6.66	0.01x + 7.4	0.002x + 7.8	0.005x + 7.5	0.004x + 13	0.005x + 11.25

Tableau 2 : Modèle pratique déduit d'une expérimentation

A partir du modèle énergétique présenté ci-dessus, différentes observations initiales peuvent être effectuées.

Par exemple :

- pour les données de petite taille, EDGE consomme moins que les interfaces WiFi et 3G
- pour les données de taille importante, WiFi est plus efficace en termes de consommation d'énergie
- la phase d'association à un réseau WiFi est très pénalisante en cas de données de petite taille

Nous reviendrons sur ce tableau et ces observations lors de la définition des règles d'optimisation.

time#185140$Datasize#7160.0$States#up$ServicesEnCours#2&service_type#VideoStreaming

Figure 42 : Exemple de contexte dynamique concernant un service VideoStreaming

Dans ce cas d'études nous avons gardé Ubiquity-Ont telle qu'elle est. En effet, pour ECO-WISM la plupart des modifications sont au niveau de l'*Avatar* utilisateur/équipement et au niveau des échanges des informations de contexte dynamique entre l'Avatar et l'entité représentée. Les seules petites modifications au niveau de l'ontologie Ubiquity-Ont sont l'enrichissement de la description des services VidéoStreamin, téléchargement de e-mail et Téléchargement de fichier. Avec cette petite adaptation, un Avatar peut estimer la taille en ko d'un service Vidéo-Streaming de durée N s et de qualité Q.

III.3.2. Mécanisme de sélection d'interface

Nous proposons un mécanisme qui vise à automatiser la sélection d'interface basé principalement sur deux critères qui sont la qualité de service et la consommation d'énergie.

L'optimisation globale de la consommation d'énergie doit minimiser deux grandeurs:

- la consommation énergétique en mode veille (idle) et
- la consommation d'énergie en mode transmission.

Le mécanisme proposé est alors composé de deux processus principaux:

- *The Session Monitoring Process*: vise à minimiser le coût énergétique en mode veille.
- *The Service Monitoring Process* : responsable de l'optimisation de l'énergie en cours de transmission.

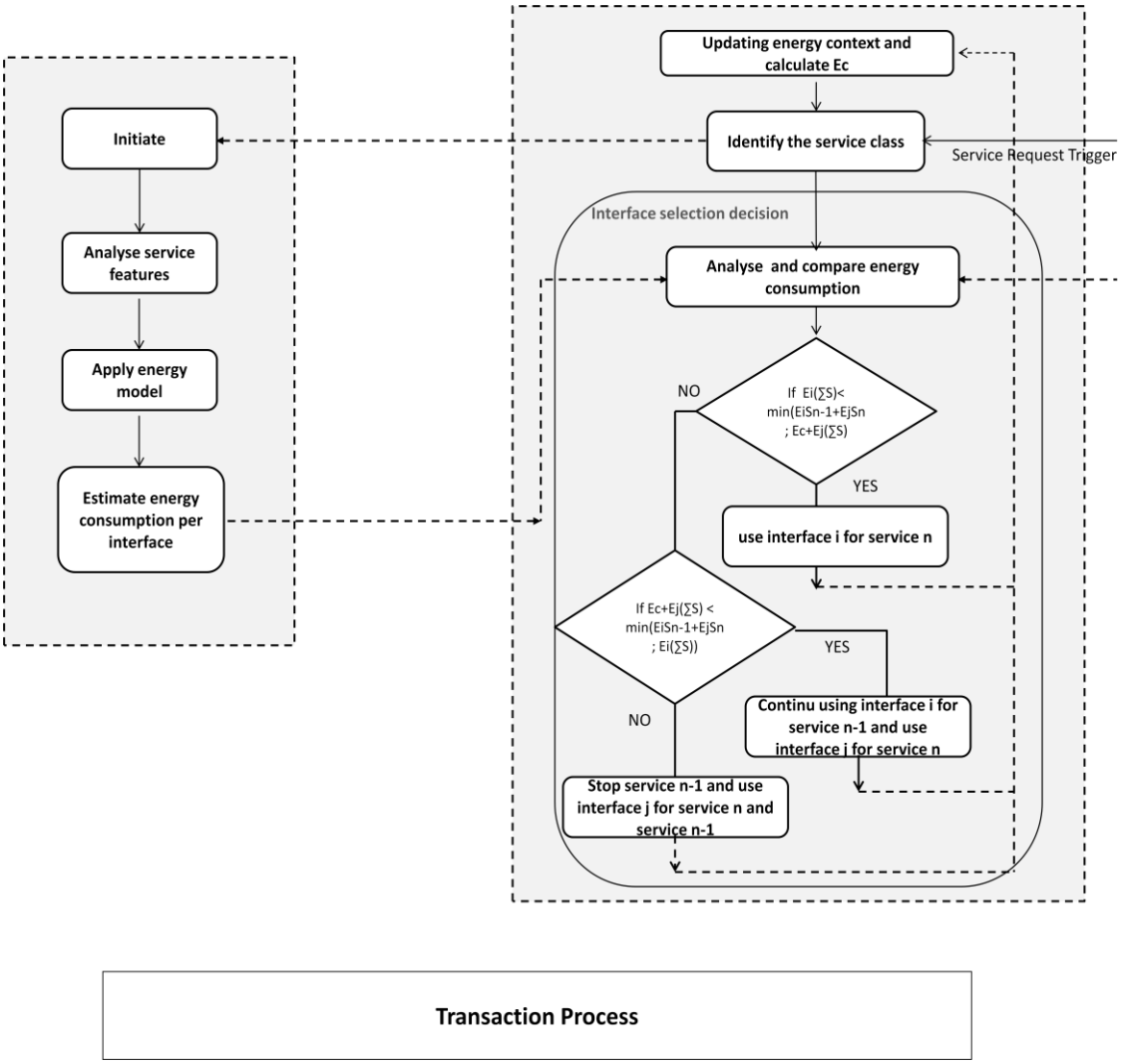


Figure 43 : Service Monitoring Process

Ces deux processus seront intégrés à l'architecture afin de prendre des décisions globales de sélection d'interface (c.f. Figure 45).

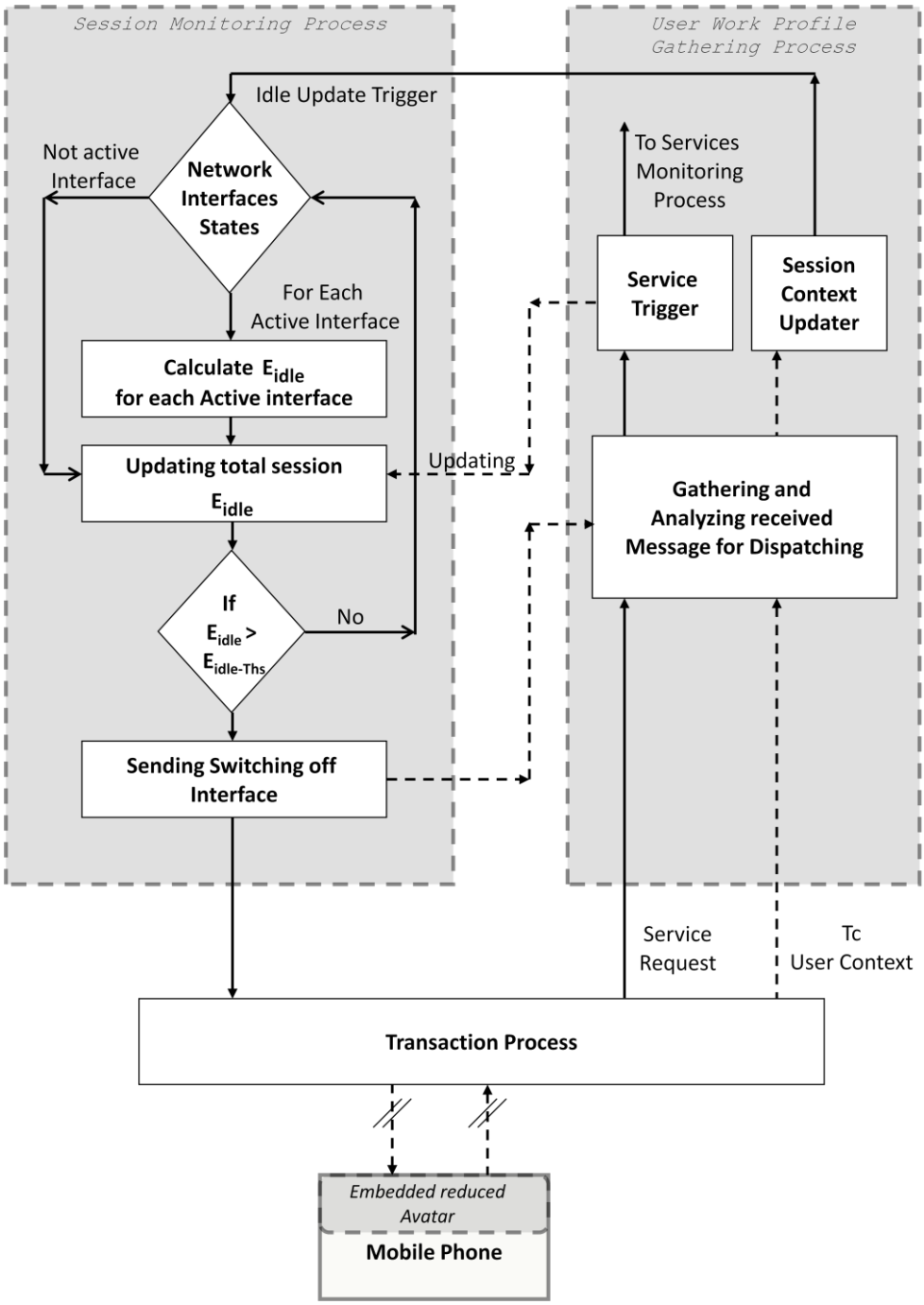


Figure 44 : « Gathering Context module » et « session monitoring process »

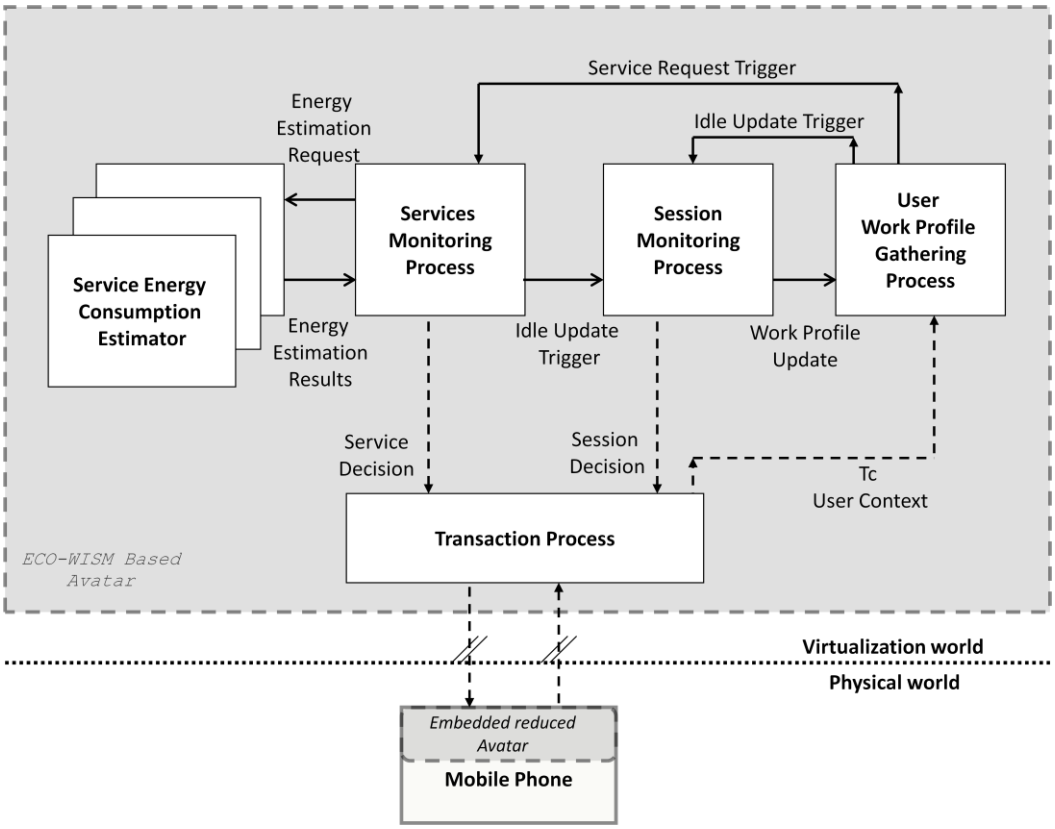


Figure 45 : Architecture de l'*Avatar* fonctionnant avec ECO-WISM

Le comportement d'un agent utilisateur est à la fois cyclique et événementiel. En effet, après chaque tcc (*time for checking context*) qui est un « *timer* » prédéfini, l'*Avatar* (qui est une machine à états) revient à l'état initial qui est « *context gathering & updates* » afin de mettre à jour son « *Work Profile* » surtout avec les informations contextuelles dynamiques (c.f. Figure 45). Par contre, ce même processus reste à l'écoute en parallèle pour recevoir des informations de contexte urgentes.

Une phase d'authentification est effectuée au démarrage de la communication. En effet, le mobile envoie à l'agent les informations de l'abonnement, ce dernier lance une requête d'authentification avec le gestionnaire de contexte global.

L'agent logiciel communique avec le deuxième niveau de l'architecture (2ème niveau d'abstraction de l'architecture présentée dans le chapitre 4) pour échanger les informations de contexte statiques. C'est informations sont essentiellement des données d'authentification (au début de la session de notre système), des données statiques décrivant le profil utilisateur

(profil, préférences et habitudes), celles décrivant les équipements, les services et les réseaux. L'ensemble de ses informations forment le workprofile de l'agent logiciel. Avec ce workprofile, l'agent est capable, après son analyse, de prendre des décisions et les envoyer (pour exécution) à l'entité représentée (mobile phone).

L'échange des informations dynamiques entre l'entité représentée (mobile phone) et l'agent. Le sens habituel c'est du mobile phone vers l'agent. L'entité envoi les informations dynamiques à l'agent pour qu'il lui soustraite son traitement et prend la décision à sa place (alléger le mobile, à ressources limitées, et faire des traitements plus compliqués). L'autre sens est aussi possible, qui consiste en fait à l'envoi des décisions prise par le bloc décisionnel (l'algorithme ECO-WISM : Energy Consumption Optimization for Wireless Interface Selection Mechanism).

Le « user *Work Profile* gathering process » analyse le « *Work Profile* » et raisonne dessus pour choisir quel bloc de Décision et Notification à choisir. Il déclenche en effet des trigger qui lancent eux-mêmes des Thread qui s'exécutent en parallèle au fonctionnement normal et cyclique de ces blocs. En effet, cette architecture est inspirée des architectures événementielles. En effet, le processus sélectionne pour chaque processus de prise de décision les informations qui ce dernier s'y abonne. Pratiquement nous avons des processus qui s'exécutent en parallèle et cycliquement et le processus user « workprofile gathering process » rempli les objets communs que d'autres processus déjà en exécution sont en train de les traiter.

Les blocs « Session Monitoring process» consiste en un processus qui s'exécute au démarrage de la session et qui peut lancer plusieurs autres processus fils à chaque mise à jour du workprofile. Ce processus supervise les interfaces et tout au long d'une session c'est lui qui décide d'éteindre une interface car il estime qu'elle a trop consommé en état IDLE pour minimiser la consommation globale d'énergie. Il peut même décider de terminer la session avec des messages « sessions switch off » si la consommation d'énergie en état IDLE est importante et a dépassé un seuil qu'on a défini.

IV. Implémentations et Tests de Performances

IV.1. Plateforme de développement

Pour évaluer le système proposé, nous avons développé un test-bed (plateforme réelle) en laboratoire. Ce test-bed comporte deux parties : l'une exécute les deux niveaux d'abstraction de l'architecture et l'autre simule le fonctionnement des utilisateurs (*mobile nodes*). Nous décrivons ci-dessous les détails de ce test-bed.

- Les deux niveaux d'abstraction, le concept de virtualization :
    - Cette partie de la plateforme est composée de deux serveurs (processeur Intel Xeon 2.8 GHz, Ram 3 GB, Mémoire Cache 4 MB, JVM 1.6, Mémoire allouée à Java égale à 512 MB, version JADE 3.4).
    - Le premier serveur héberge le premier niveau d'abstraction : la plateforme qui exécute le fonctionnement des agents des utilisateurs et ceux des réseaux. Chaque agent porte le nom (ID) de son entité correspondante (MN ou Network). Chaque agent utilisateur se lance à la demande du MN et analyse le dynamic-context reçu régulièrement par ce dernier et décide à sa place s'il faut faire un HO et vers quel réseau. Les agents utilisateurs communiquent avec les agents réseaux selon le fonctionnement du mécanisme de gestion de la réputation (mise à jour, interrogation, …).
    - Le deuxième serveur héberge le 2ème niveau d'abstraction : le gestionnaire de contexte statique et profils des utilisateurs et des réseaux. Cette plateforme consiste en une base de données sémantique avec son gestionnaire et un agent software qui la représente et qui communique avec le 1er niveau d'abstraction (i.e. les agents utilisateurs et ceux des réseaux).
    - Les deux plateformes sont dynamiques et configurable. Elles sont développées en Java orientée objets.

- L'émulation du fonctionnement des « mobile nodes »
    - Pour évaluer les performances de la plateforme et la mettre à l'échelle il fallait faire plusieurs scénarii durant lesquels nous varions le nombre des réseaux disponibles,

leurs technologies, leurs dispositions et couvertures, le nombre des utilisateurs dans le système ainsi que la mobilité de ces derniers. Pour ce faire, nous avons simulé pour chaque scénario (fixer le nombre de réseaux et leurs technologies et faire varier le nombre d'utilisateurs, puis varier le nombre des réseaux disponible à chaque fois) le modèle de mobilité pour chaque utilisateur. Le modèle de mobilité utilisé est GAUSS-MARKOV.

- Ainsi nous avons obtenu des fichiers traces pour chaque utilisateur. Ces fichiers contiennent plusieurs lignes, et chaque ligne représente une itération (chaque 3 secondes).
- Pour émuler le fonctionnement du MN nous avons développé des agents logiciels (qui représentent l'agent light à embarquer dans le MN). Ces derniers envoient chaque lapse de temps (3 seconde) le dynamic-context à son agent virtuel (Avatar) qui le représente dans le niveau d'abstraction. Le dynamic-context contient l'ensemble des réseaux disponibles ainsi que le détail du réseau sur lequel le MN est normalement connecté (BER, jitter, ...).

IV.2. 1er scénario d'expérimentation : SEAMLESS

Le gestionnaire de contexte utilise une ontologie qui décrit l'ensemble de l'environnement à savoir les clients, leurs terminaux, les capacités de ces derniers, les services et leurs exigences et les relations entre ces concepts. Cette description étant abstraite nous avons crée des instances pour valider l'architecture proposée dans le premier cas d'étude :

- Le service considéré : Caméra de surveillance IP
- Plusieurs clients qui ont des terminaux différents
- Un client (Bob) demande le service depuis son terminal et notre système détecte à partir de son contexte que les capacités de son équipement sont inférieures à celles exigées par le service vidéo.
- Une instance d'un transcodeur et lancée pour cet utilisateur pour adapter la vidéo à son contexte.
- D'autres clients visualisent la vidéo sans adaptation puisqu'ils ont des terminaux avec des capacités supérieures que celle de Bob et en ligne avec les exigences du service.

IV.3. Mesures de performances

Les mesures effectuées sur la plateforme sont très encouraants sur les aspects fonctionnels et de performances.

La Figure 46 montre une comparaison des temps de réponse de deux mécanismes de handover vertical : celui basé sur le mécanisme de réputation et la gestion de contexte et celui de Savitha et al [5] que nous considérons ici comme mécanisme de référence.

Les résultats montrent que malgré l'utilisation des données contextuelles, le mécanisme de handover vertical basé sur la notion de réputation ne souffre pas de délais excessifs, qui sont même plus réduit dans ce scénario.

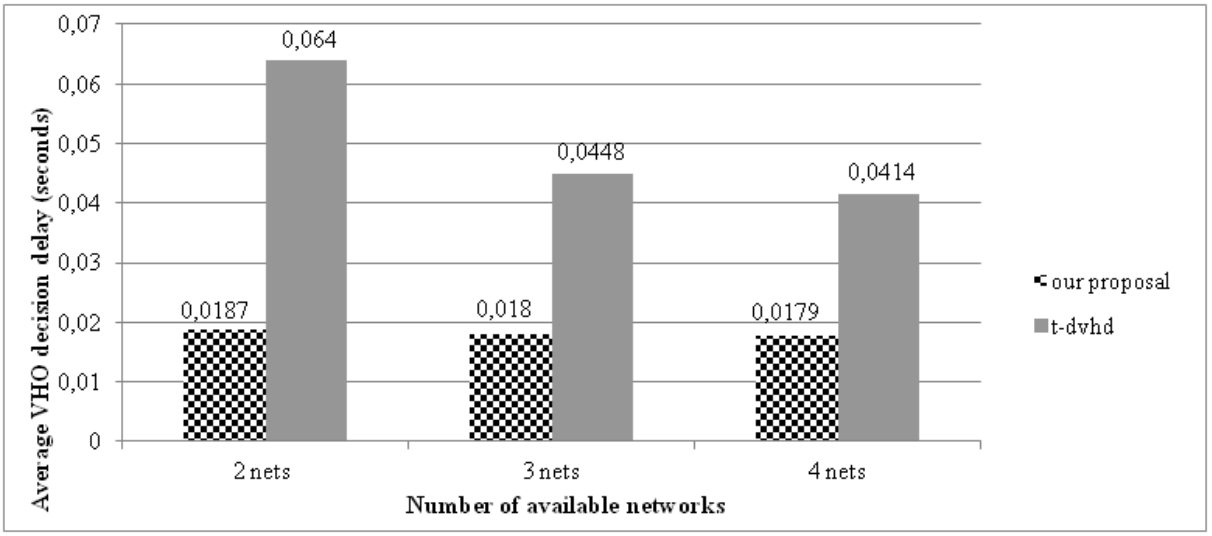


Figure 46 : Comparaison des performances de notre proposition à celle du t-dvhd

La Figure 47**Erreur ! Source du renvoi introuvable.** montre le temps moyen nécessaire aux décisions de handover quand le nombre d'utilisateurs effectuant simultanément un handover augmente. Quand ce nombre est entre 5 et 80 et pour 12 réseaux disponibles, le délai moyen de décision varie entre 58 et 76 millisecondes.

Ce délai est entre 15 et 29 millisecondes quand 4 réseaux uniquement sont disponibles. Ce délai est également peu sensible au nombre d'utilisateurs simultanés.

Sur la Figure 48, nous présentons les délais moyens de décision, pour 20 utilisateurs simultanés et en faisant varier le nombre de réseaux disponibles.

Les résultats montrent que ces délais restent inférieurs à 30 millisecondes quand le nombre de réseaux est inférieur à 8. Ce délai augmente ensuite pour atteindre 58 millisecondes pour 12 réseaux. Ceci s'explique par l'accroissement des données contextuelles qu'il faut échanger et analyser concernant les différents réseaux disponibles. Ces échanges concernent essentiellement les agents utilisateurs et les agents représentant les réseaux.

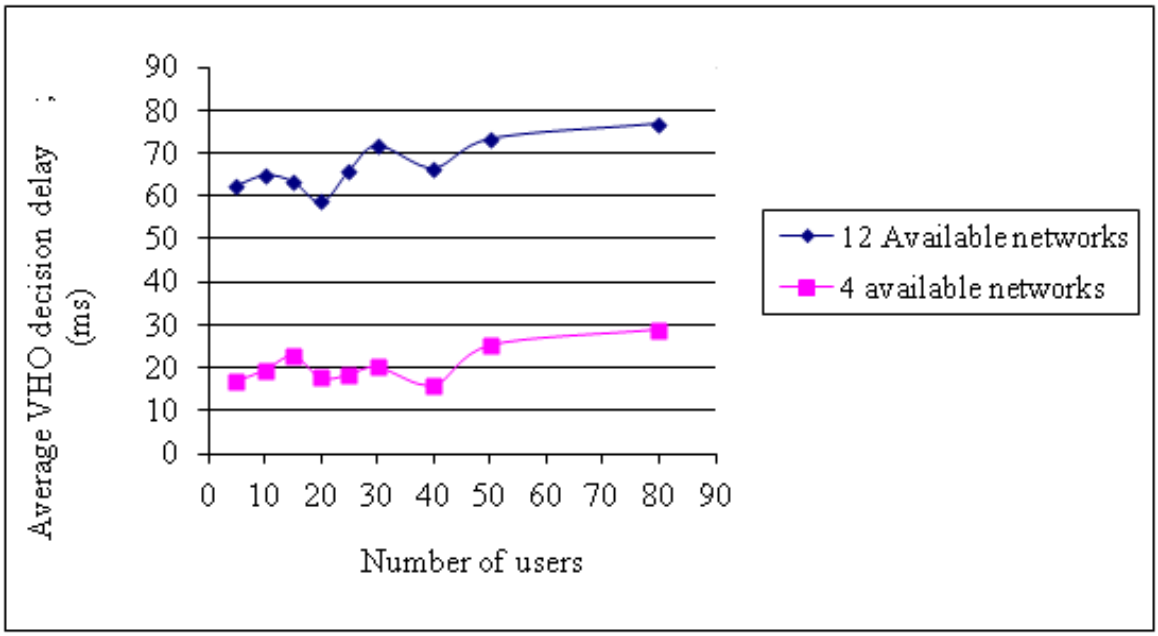


Figure 47 : Comparaison des délais moyens de prise de décisions entre 4 et 12 réseaux disponibles en fonction du nombre des utilisateurs

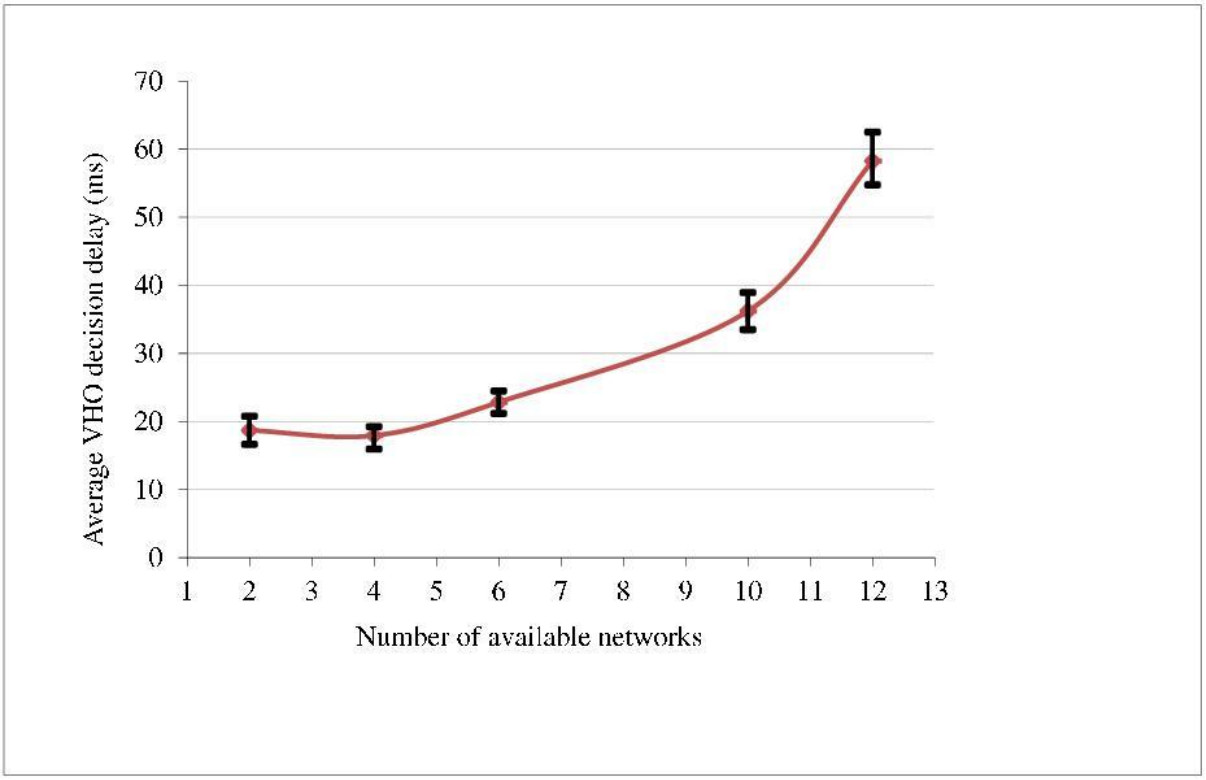


Figure 48 : Délais moyens de prise de décision en fonction du nombre des réseaux disponibles pour 20 utilisateurs

IV.4. 2er scénario d'expérimentation : ECO-WISM

Pour ce deuxième scénario, nous avons choisi le scénario suivant :

- Invocation de services aléatoire
- Types de services invoqués :
  - vidéo streaming (de qualité et durée aléatoires),
  - téléchargement de fichier,
  - téléchargement de e-mail

Notre objectif est d'étudier le comportement de la proposition ECO-WISM d'un coté, et d'autres cotés d'étudier l'influence de la façon d'invoquer les services et leur distribution tout au long d'une session sur la consommation d'énergie. Nous avons alors comparé la proposition ECO-WISM à un modèle de sélection d'interface basé uniquement sur le niveau de signal.

Les résultats de mesure sur la plateforme montrent que la proposition ECO-WISM influe positivement sur l'économie de la consommation de l'énergie avec un pourcentage de gain qui peut atteindre 35% dans certains scénarios. Ce gain est dû au fait que ECO-WISM gère l'activation et la désactivation des interfaces réseaux tout en surveillant (monitorant) ces dernieres lorsqu'elles sont en mode IDLE d'un coté, et d'en choisir une pour un service donnée lorsque notre algorithme de sélection pense qu'elle est la meilleure en terme de QoS et de consommation d'énergie.

L'algorithme ECO-WISM a été développé avec JAVA comme langage de développement

La Figure 49 montre que les choix en termes d'interfaces réseaux ECO-WISM, avec les règles et les modèles de consommation d'énergie, respectent bien les exigences des services en termes des qualités de services requises. Par exemple, on peut vérifier que les services gourmandes en terme de bandes passantes, tel que le vidéo streaming, passent souvent sur le réseau Wifi, par contre les services qui n'exigent ce type de qualité minimale, tel que la réception d'un e-mail de quelques kilos octets de taille, passent la plupart des temps par le réseau cellulaire EDGE. Tout en respectant son objectif, minimiser la consommation d'énergie ECO-WISM respecte bien le besoins des services en termes de QoS.

Pour évaluer notre proposition nous avons exécuté notre simulateur avec et sans notre mécanisme ECO-WISM. Les résultats que nous avons obtenus montrent que l'utilisation de notre algorithme diminue la consommation de l'énergie. En effet, tout au long d'une session, la consommation d'énergie par interface réseau, soit lorsqu'elle est en Idle ou bien en utilisation par un service, et du coup la consommation totale du terminal client par l'ensemble de ses interfaces réseaux est réduite par un pourcentage de 20% par rapport à une utilisation du terminal sans ECO-WISM. La figure 4 montre bien les courbes de consommation d'énergie en Joules en fonction du temps avec et sans ECO-WISM.

L'histogramme illustré par la figure 5 montre qu'avec notre proposition augmente le temps durant lequel les interfaces sont Down et du coup le terminal client consomme beaucoup moins d'énergie. En effet, ECO-WISM n'optimise seulement pas l'utilisation des interfaces tout en choisissant la ou les meilleures interfaces réseaux pour un service donné, mais aussi il suit chaque interface durant la session pour diminuer la consommation de cette dernière même lorsqu'elle est en mode Idle.

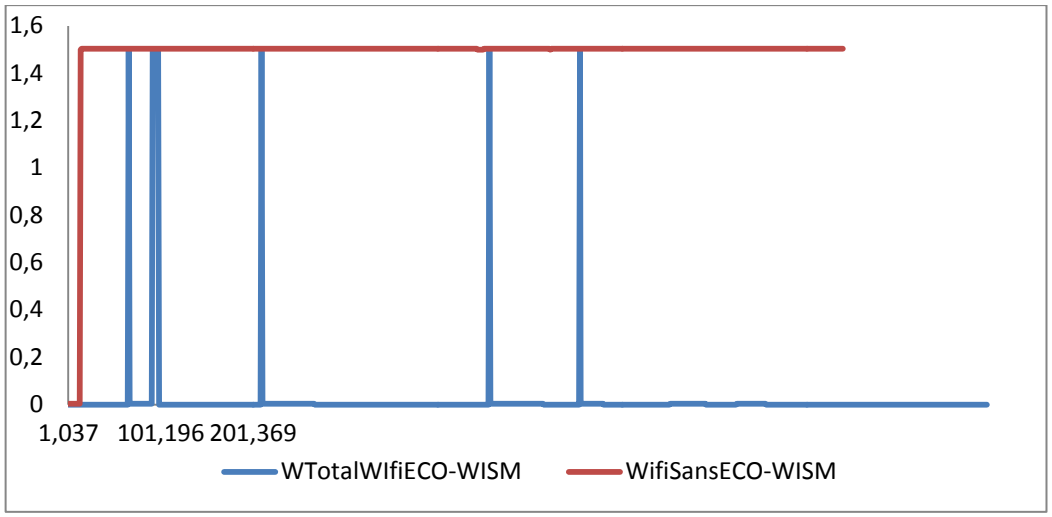


Figure 49 : Courbe de consommation d'énergie (en J) de l'interface WiFi en fonction du temps (en s) avec et sans ECO-WISM

L'analyse des performances de cette proposition a été effectuée en émulant le comportement des utilisateurs en ce qui concerne l'invocation des services. La solution ECO-WISM a été développée en JAVA.

Nous avons considéré que chaque utilisateur peut invoquer aléatoirement des services de différentes classes tel que le vidéo streaming, le transfert de fichiers et de courriels.

Nous avons considéré que ces services sont invoqués dans le cadre de sessions. Une session est définie par la période de temps pendant laquelle le terminal de l'utilisateur est actif pour au moins un service. Le nombre et les durées des services invoqués par chaque utilisateur sont aléatoires.

La solution proposée est ensuite évaluée en termes de consommation d'énergie et comparée à un système de référence sans mécanismes spécifiques pour l'optimisation énergétiques.

Les résultats obtenus nous permettent de conclure que la solution ECO-WISM atteint bien ses objectifs et permet de réduire de façon, parfois conséquente, la consommation d'énergie au niveau des terminaux. Les gains enregistrés peuvent aller jusqu'à environ de 40% en fonction des scénarios considérés.

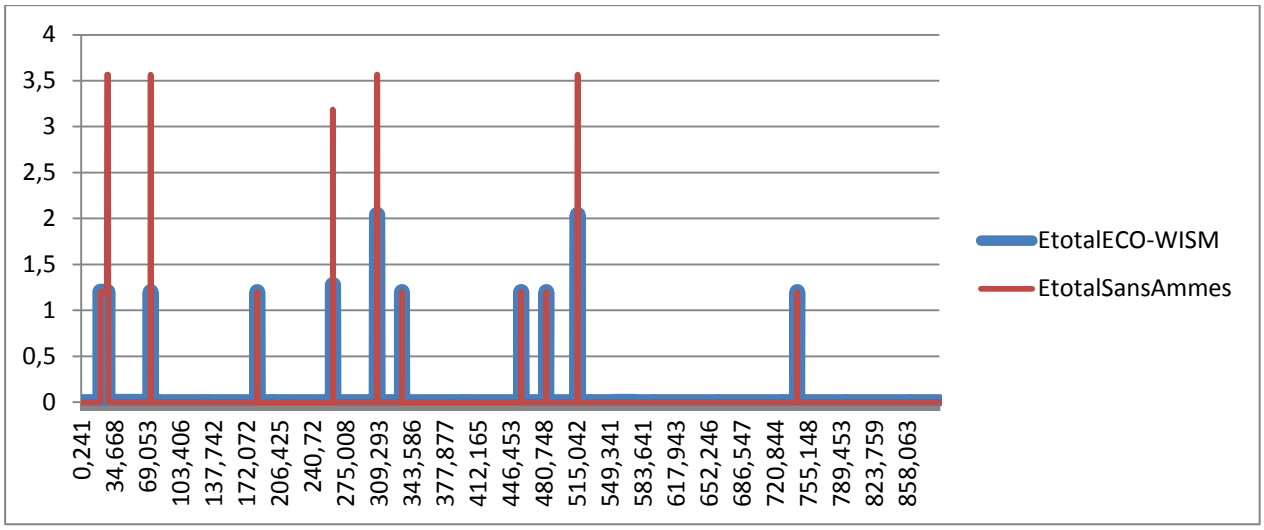


Figure 50 : total de consommation d'énergie (en J) avec et sans ECO-WISM en fonction du temps (en s)

Ce gain est dû au faite que ECO-WISM gère l'activation et la désactivation des interfaces réseaux tout en monitorant ces derniers lorsqu'elles sont en mode IDLE d'une coté, et de les choisir pour un service donnée lorsque notre algorithme trouve qu'elles sont les meilleures pour consommer moins d'énergie.

La figure 3 montre que les choix en termes d'interfaces réseaux ECO-WISM, avec les règles et les modèles de consommation d'énergie, respecte bien les exigences des services en termes des qualités de services requises. Par exemple, on peut vérifier que les services gourmandes en terme de bandes passantes, tel que le vidéo streaming, passent souvent sur le réseau Wifi, par contre les services qui n'exigent ce type de qualité minimale, tel que la réception d'un e-mail de quelques kilos octets de taille, passent la plupart des temps par le réseau cellulaire EDGE. Tout en respectant son objectif, minimiser la consommation d'énergie ECO-WISM respecte bien les besoins des services en termes de QoS.

Pour évaluer notre proposition nous avons exécuté notre simulateur avec et sans notre mécanisme ECO-WISM. Les résultats que nous avons obtenus montrent que l'utilisation de notre algorithme diminue la consommation de l'énergie. En effet, tout au long d'une session, la consommation d'énergie par interface réseau, soit lorsqu'elle est en Idle ou bien en utilisation par un service, et du coup la consommation totale du terminal client par l'ensemble de ses interfaces réseaux est réduite par un pourcentage de 20% par rapport à une utilisation du

terminal sans ECO-WISM. La figure 4 montre bien les courbes de consommation d'énergie en Joules en fonction du temps avec et sans ECO-WISM.

L'histogramme illustré par la figure 5 montre qu'avec notre proposition augmente le temps durant lequel les interfaces sont Down et du coup le terminal client consomme beaucoup moins d'énergie. En effet, ECO-WISM n'optimise seulement pas l'utilisation des interfaces tout en choisissant la ou les meilleures interfaces réseaux pour un service donné, mais aussi il suit chaque interface durant la session pour diminuer la consommation de cette dernière même lorsqu'elle est en mode Idle.

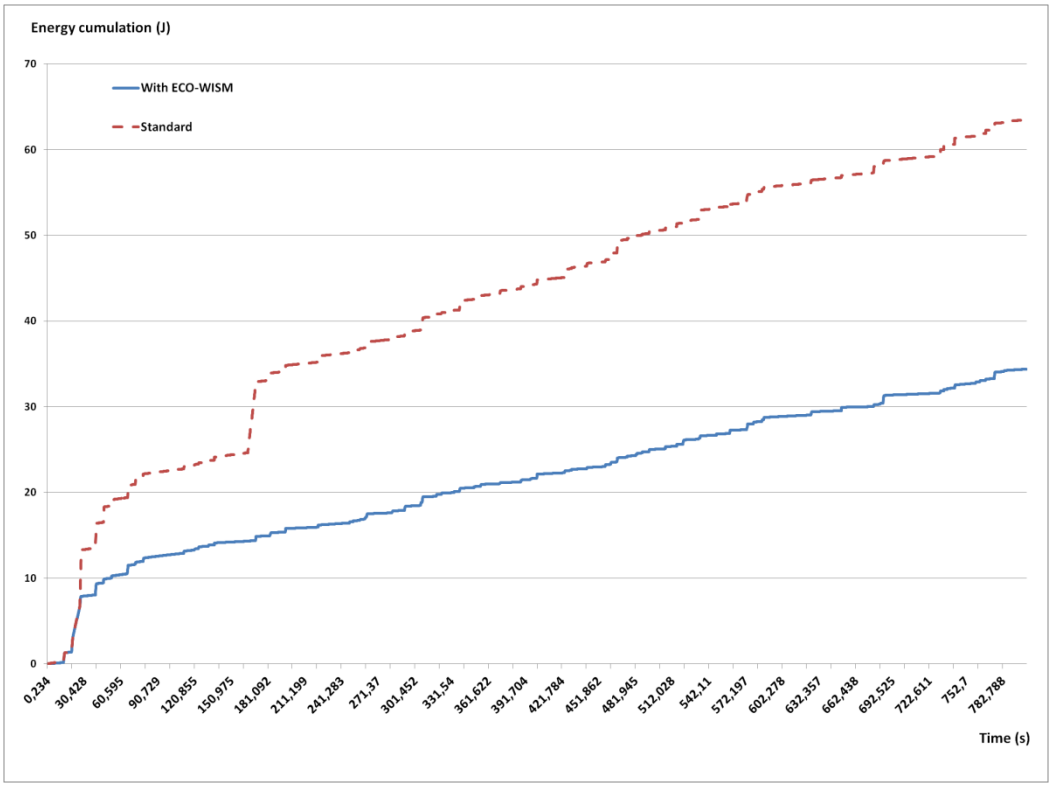


Figure 51 : Cumul de consommation d'énergie (en J) avec et sans ECO-WISM en fonction du temps (s)

V. Conclusion

Dans ce chapitre, nous avons confronté nos différentes propositions : architecture de et ontologie à deux problématiques réelles : la gestion de la mobilité sir des environnements de mobiles. Les deux cas d'étude offrent des environnements riches et assez complexes.

Les résultats obtenus montrent que nos propositions peuvent parfaitement être appliquées à ces cas et qu'elles offrent des résultats très encourageant en termes fonctionnel et de performances.

Grace aux concepts et mécanismes proposés, nous avons en effet pu introduire une architecture capable d'introduire la gestion de contexte sur des environnements existant, grâce à la virtualisation. Les deux cas d'étude considérés ont permis (a) de valider nos propositions en termes fonctionnels et de performances et (b) ce compléter la méthodologie proposée grâce à une analyse de type BottomUp.

.

Chapitre 6 : Conclusion générale et perspectives

La conclusion de ce rapport est le moment pour prendre du recul sur le travail accompli jusqu'aujourd'hui. Le premier et le deuxième chapitre ont été consacrés pour introduire le sujet et étaler les travaux similaires établis dans le même domaine de recherche.

La conception d'un système permettant d'avoir un environnement ubiquitaire nécessite la prise en considération de l'hétérogénéité de ce dernier. Un système ouvert nous oblige à distribuer les fonctionnalités. L'utilisation des systèmes multi-agents répond bien à cette exigence mais aussi offre l'autonomie aux fonctionnalités et composants du système. Pour évaluer notre système proposé il faut que nous définissions des critères d'évaluation et des mesures intensives pour montrer la mise en échelle. Parmi ces critères, nous travaillons sur les délais des réponses et les délais d'échanges (les communications entre les deux niveaux et les communications entre agents ainsi que le raisonneur de l'ontologie). D'une autre côté l'avantage de l'utilisation d'une ontologie c'est l'enrichissement sémantique des informations contextuelles. Dans nos travaux courants nous travaillons sur la conception d'une ontologie générique ouverte et extensible.

Le travail que nous avons présenté ici porte une attention particulière aux propriétés suivantes des infrastructures de gestion d'information de contexte qui nous semblent indispensables pour la construction d'applications d'informatique diffuse :

- **L'ouverture** est la première exigence que nous nous sommes imposés dans notre problématique introduisant l'environnement hétérogène des TIC. L'utilisation des technologies du web sémantique à savoir l'ontologie *Ubiquity-Ont* que nous avons proposé, dans notre architecture lui permet d'accepter l'introduction de composants, d'équipements, de services et d'applications inconnus et imprévus pendant l'exécution, sans arrêt ou reconfiguration. L'utilisation de technologies standards encourage l'intégration de nos propositions dans les applications déjà existantes (il faut qu'elles soient ouvertes).
- **Le dynamisme**, ou **la flexibilité** : prendre en compte l'hétérogénéité des équipements, des applications et aussi bien des représentations de contexte pendant l'exécution. Ceci est une nouvelle fois rendu possible à l'aide des ontologies et de l'introduction de la

sémantique dans la représentation des informations contextuelles. Ceci est notamment grâce à la possibilité de toujours étendre une représentation par une autre représentation plus objective et facilitant le dynamisme du système.

Dans [28] et comme nous l'avons indiqué dans le chapitre 2, une représentation des informations contextuelles doit respecter six critères. Ces critères sont toutes satisfaits par l'utilisation des technologies du web sémantique, notamment RDF ou OWL dans notre architecture avec l'intégration de *Ubiquity-Ont*.

D'un autre côté les systèmes et infrastructures de gestion de contexte doivent respecter certaines exigences présentées dans [80] et [18] par exemple. Nous n'avons pas directement considéré toutes ces exigences dans nos travaux, mais dans les cas d'études que nous avons considérés notre solution peut effectivement répondre à ces exigences :

- **Mobilité** : nous avons fourni des dispositions et mécanismes pour supporter la mobilité des équipements. En effet, avec l'intégration de plusieurs variétés des algorithmes proposés par Zekri et al. [77] et [78], l'implémentation de nos solutions en utilisant une plateforme d'exécution multi-agents (une implémentation de la notion des *Avatars* que nous avons définis dans le chapitre 3) supporte la mobilité des équipements, l'adressage et le transport de messages d'une plateforme à une autre plateforme que nous avons définis comme étant « zones actives » (géographiques et logiques).
- **Scalabilité** : Nous n'avons pas assez testé notre infrastructure pour prouver son passage à l'échelle dans un environnement ouvert réel. Cependant la réalisation du démonstrateur a déjà prouvé que l'on peut déployer un nombre non négligeable d'applications et d'équipements en utilisant notre infrastructure.
- **Tolérance aux pannes** : Le seul composant centralisé est le deuxième niveau d'abstraction gérant les informations de contexte statiques. Nous n'avons pas vraiment étudié la résistance de notre infrastructure aux pannes. Cependant la redondance de ce niveau permettrait de surmonter cette faiblesse, et cela est possible avec plusieurs techniques de redondance existantes. D'un autre coté, l'ouverture et le caractère distribué du premier niveau sont une bonne base pour obtenir un système tolérant aux pannes.
- **Sécurité, traçabilité et contrôle** : Ceci est introduit par la notion de gestion de l'historiques (c.f. chapitre 3). Nous avons introduit cette notion et nous avons proposé de simples mécanismes et techniques pour l'implémenter, surtout dans le cas d'étude du

projet ANR/SEAMLESS, mais les développeurs peuvent facilement étendre ces techniques ou bien utiliser d'autres librairies pour stocker des descriptions OWL et RDF dans des instances de notre base de données sémantique ou bien de faire des liens avec des bases de données relationnelles habituelles. D'un autre côté, le flux d'information circulant entre les producteurs d'information de contexte et les consommateurs est caché des utilisateurs. Par contre, nous n'avons pas étudié les aspects de sécurité tout au long de nos travaux et propositions d'implémentations, mais l'ouverture de nos propositions permettra l'intégration des standards existants et donner la possibilité aux chercheurs de proposer de nouvelles techniques et mécanismes de sécurité et de respect de la vie privée.

La gestion de contexte dans le domaine des TIC a été introduite avec l'avènement de l'intelligence artificielle et récemment reprise avec les notions d'informatique diffuse et d'informatique omniprésente. Elle sera la base des systèmes pervasifs et autonomiques de demain.

L'idée est de contrer la complexité des systèmes de plus en plus étendus et hétérogène, par l'introduction d'une certaine forme d'intelligence et d'autonomie. On vise alors des systèmes auto-gérés ou auto-adaptatifs, capables de se reconfigurer en fonction de leurs contextes.

Mais à cause du volume et de la complexité des connaissances à traiter, les techniques traditionnelles de gestion de données peuvent difficilement offrir les outils permettant à l'introduction de l'intelligence nécessaires à ces futurs systèmes.

L'introduction de la représentation sémantique pour représenter les données de contexte est une première étape pour ces systèmes. Elle permettra de représenter l'environnement technologique par des bases de données sémantiques sur lesquelles il sera possible de développer des raisonneurs automatisés pour la prise de décision dans les systèmes autonomiques. Ceci peut être effectué par différents langages tels que RDF. Les ontologies offrent un autre niveau de représentation puis elles permettent du raisonnement sur les connaissances.

D'un autre coté, l'introduction de la sémantique et du raisonnement sur les entités d'un système de télécommunication n'est pas une tâché aisée. La complexité des traitements engendrés par ces notions peu alourdir le fonctionnement des entités du système. Dans certains cas, ces traitement sont purement et simplement inimaginables tel que sur les terminaux mobiles, limités en ressources et en énergie.

La disponibilité d'une ontologie pour le domaine des TIC semble donc une nécessité afin d'accompagner les nouvelles tendances vers l'informatique ubiquitaire et les réseaux ambiants.

Même si plusieurs tentatives existent dans la littérature, la plupart des solutions proposées sont centrées sur des sous domaines particuliers tels que les services ou les terminaux mobiles.

La construction d'une ontologie générique, extensible et ouverte pour le domaine des TIC nous semble une priorité pour la communauté de recherche. Dans cette thèse, nous n'avons pas visé la conception d'une ontologie complète pour le domaine des TIC. Ceci prendrait des dizaines d'années puisqu'il faudra identifier, structurer et décrire sémantiquement l'ensemble des entités logiques et physiques des TIC et de leurs interactions.

Plusieurs problématiques rendent difficile aujourd'hui de proposer de vrais systèmes autonomes et context-aware :

- Manque d'outils génériques pour la représentation sémantique
- Manque d'outils génériques pour la collecte et le partage
- Les informations de contexte peuvent être incomplètes, indisponibles ou non mises à jour
- Complexité des opérations liées à ces données sémantiques et difficulté de les traiter sur toutes les entités du domaine des TIC

Dans cette thèse, nous avons abordé ces différentes problématiques du point de vue conceptuel pour proposer de nouvelles solutions et approches, que nous avons ensuite confrontées à deux cas d'études liés à la gestion de mobilité sur des environnements de réseaux d'accès hétérogènes et à l'optimisation de la consommation d'énergie.

Après le premier chapitre consacré à l'analyse de l'état de l'art sur la gestion de contexte, nous avons dans un premier temps proposé une méthodologie de conception pour une ontologie générique, ouverte et extensible dans le domaine des TIC.

Nous avons préféré concevoir une méthodologie pour la construction graduée d'une telle ontologie. Cette méthodologie est itérative et composée de sept étapes. Elle mixe les approches TopDown et BottomUp permettant d'affiner l'ontologie et de la vérifier du point de vue conceptuel et pratique. Cette méthodologie a ensuite été utilisée pour générer une première ontologie pour le domaine des réseaux et services de télécommunications.

Afin de décharger les entités physiques des traitements liés à la gestion de contexte, et afin de permettre à des équipements hétérogènes de coopérer autour d'une même ontologie, nous avons ensuite proposé une solution overlay composée de deux niveaux de virtualisation. Elle permet de superposer à l'environnement physique, des composants logiques responsables de la manipulation des données de contexte et de leur gestion. Ces composants pourront alors utiliser les ontologies pour raisonner tout en déchargeant les entités physiques de traitements complexes liés à la gestion de contexte.

Le choix de deux niveaux de virtualisation découle du niveau de dynamicité des informations de contexte considérées et du besoin d'avoir une vue globale et d'un niveau d'orchestration.

Le premier niveau est le plus actif car proche des entités gérées. Il manipule les informations de contexte les plus dynamiques provenant des entités de l'environnement. Il repose sur des composants logiques que nous avons appelés *Avatars*, qui se substituent aux entités physiques dans l'espace virtuel.

Le deuxième niveau, possède une vue globale et gère les informations contextuelles, peu dynamiques voire statiques. Il est responsable de l'orchestration des différents composants de l'architecture. Il utilise un gestionnaire de contexte basé sur la sémantique et les ontologies pour la description de l'environnement et de ses éléments contextuels. Un module de raisonnement et d'inférence permet à ce plan de raisonner et de prendre des décisions autonomes.

L'architecture proposée inclut également des aspects non fonctionnels, tels que la mobilité des Avatars et la notion de Zones actives, qui peuvent être physiques et logiques. Ces notions permettent d'optimiser le fonctionnement du plan overlay et rendent la solution plus flexible et plus extensible.

La mobilité des Avatars, permet par exemple de rapprocher les composants de virtualisation des entités qu'ils représentent et ainsi d'optimiser les délais d'échanges des données contextuelles. Les *zones actives* permettent quant à elles de raisonner par domaine ou par service en regroupant de façon logique les entités de chaque domaine par zone.

L'ontologie et l'architecture proposées dans cette thèse ont été validées sur une plateforme réelle pour la gestion de contexte des réseaux et services hétérogènes. Nous les avons appliquées à deux cas d'études liés à la gestion de mobilité sur des environnements de réseaux d'accès hétérogènes et à l'optimisation de la consommation d'énergie. Ces deux cas d'étude ont

permis (a) de valider nos propositions en termes fonctionnels et de performances et (b) de compléter la méthodologie proposée en permettant de compléter l'ontologie proposée par l'approche BottomUp.

Bibliographie

[1] M. Weiser, “The computer for the twenty-first century,” vol. 3, no. 256, pp. 94-104, 1991.

[2] A. K. D. a. G. D. Abowd, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” vol. 16, p. 97–166, 2001.

[3] P. Dourish, “Seeking a foundation for context-aware computing,” vol. 16, no. 2-3, 2001.

[4] M. Chalmers, “A Historical View of Context,” vol. 13, pp. 222-246, 2004.

[5] J. P. F. R. J. Euzenat, “Gestion dynamique de contexte pour l’informatique diffuse,” p. 113, 2006.

[6] R. Guha, “Contexts : a formalization and some applications,” (Technical Report STAN-CS-91-1399- Thesis et MCC ACT-CYC-423-91), Stanford university (CA US), 1991.

[7] B. C. C. P. W. F. BANÂTRE Michel, Informatique diffuse Texte imprimé : des concepts à la réalité, vol. 1, Paris: Hermès science publ. Lavoisier, 2007, p. 204.

[8] B. Ahlgren, “Ambient networks: bridging heterogeneous network domains,” vol. 2, pp. 937-941, 2005.

[9] G. C. a. D. Kotz., “on Mobile Computing Systems and Applications,” 2002.

[10] J. C. a. G. Rey, “In ACM CADUI,” pp. 283-302, 2002.

[11] R. t. Diane Lingrand and Michel Riveill, “Input interactions and context component based modelisations: differences and similarities,” 2006.

[12] V. T. Oleksiy Khriyenko, Context description framework for the semantic web, Paris (FR): Proceedings Context 2005 Context representation and reasoning workshop, 2005.

[13] M. Satyanarayanan, “Pervasive computing: vision and challenges,” vol. 8, no. 4, pp. 10-17, 2001.

[14] D. Saha, “Pervasive computing: a paradigm for the 21st century,” vol. 36, no. 3, pp. 25-31, 2003.

[15] J. C. G. Klyne, Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, 2004.

[16] R. V. G. D. B. Lenat, “Building Large Knowledge-Based Systems ; Representation and

Inference," 1989.

[17] J. deKleer, An assumption-based TMS, X. P. A. R. C. 3. C. H. R. P. A. C. 9. Intelligent Systems Laboratory, Ed., U.S.A: Artificial Intelligence, 2003, p. 127–162.

[18] J. C. S. D. D. G. J. Coutaz, "Context is key," vol. 48, pp. 49-53, 2005.

[19] J. H. O. L. Tim Berners-Lee, The semantic web, Scientific American, 2001.

[20] R. F. R. M. R. Guha, "Contexts for the Semantic Web," vol. LNCS 3298, pp. 32-46, 2004.

[21] G. S. E. M. Dean, "OWL Web Ontology Language: Reference," 2004.

[22] V. T. O. Khriyenko, "Context Description Framework for the Semantic Web," 2005.

[23] F. G. F. v. H. L. S. H. S. P. Bouquet, "C-OWL : contextualizing ontologies," pp. 164-179, 2003.

[24] D. J. A. S. J. G. Raz, "Fast and Efficient Context-Aware Services," p. 250, April 2006.

[25] C. H., "An Intelligent Broker Architecture for Pervasive Context-Aware Systems," 2004.

[26] D. S. R. F. Baldauf M., A Survey on ContextAware Systems. International Journal of Ad Hoc and Ubiquitous Computing, Inderscience Publishers, 2006.

[27] A. Dey, "Providing Architectural Support for Building Context-Aware Applications," *Ph.D. thesis,* Nov. 2000.

[28] T. a. G. H. S. Buchholz, "Comprehensive structured context profiles (CSCP) : Design and experiences," *2nd IEEE Conference on Pervasive Computing and Communications Workshops (PERCOMW),* pp. 43-47, 2004.

[29] F. Knight, "Risk, Uncertainty and Profit," 1921.

[30] V. Govindarajan, "Appropriateness of accounting data in performance evaluation: an empirical examination of environmental uncertainty as an intervening variable," vol. 9, pp. 122-135, 1984.

[31] F. Hartman, "The appropriateness of RAPM: toward the further development theory," vol. 25, pp. 451-482, 2000.

[32] R. Herring, "Managing International Risk," 1983.

[33] R. L. R. Daft, "Organizational information requirements, media richness and structural design," vol. 32, no. 5, pp. 554-571.

[34] R. C. L. W. S. Atkinson, "Fundamental Uncertainties in projects and the scope of project

management," *International Journal of Project Management,* vol. 24, pp. 687-698, 2006.

[35] J. Galbraith, "Designing complex organizations," 1973.

[36] R. M. D. Chenhall, "The Impact of Structure, Environment, and Interdependence on the Perceived Usefulness of Management Accounting Systems," vol. 61, no. 1, pp. 16-35, 1986.

[37] J. Pierson, Une infrastructure de gestion de l'information de contexte pour l'intelligence ambiante, Thèse - Université Joseph Fourier - Grenoble 1, octobre 2009.

[38] T. Lemlouma, "Architecture de Négociation et d'Adaptation de Services Multimédia dans des Environnements Hétérogènes," 2004.

[39] M. Asadi, "Context-Aware Semantic Adaptation of Multimedia Presentations," pp. 362-365 , 2005.

[40] S. G. R. L. G. Rossi, "Génération de services dépendant du contexte pour des applications mobiles," *Actes des Premières Journées Francophones: Mobilité et Ubiquité,* 2004.

[41] R. Prasad, "My personal Adaptive Global NET (MAGNET)," no. ISBN : 978-90-481-3436-6, 2010.

[42] G. Rey, "Contexte en Interaction Homme-Machine : le contexteur," *Phd thesis,* 2005.

[43] T. S. a. C. Linnhoff-Popien, "A context modeling survey. InWorkshop on Advanced Context Modelling, Reasoning and Management," *UbiComp 2004,* 2004.

[44] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition,* vol. 5, pp. 199-220, 1993.

[45] D. a. G. R. Brickley, "Resource Description Framework (RDF) Schema Specification," vol. Proposed Recommendation, 1999.

[46] N. F. a. M. D. L. Noy, "Ontology development 101: A Guide to Creating Your First Ontology," *Development,* vol. 32, pp. 1-25, 2001.

[47] D. a. W. J. McGuinness, "Conceptual Modeling for Configuration: A Description Logic-based Approach," *Artificial Intelligence for Engineering Design,* Vols. Analysis, and Manufacturing special issue on Configuration, 1998.

[48] T. G. J. E. H. P. A. T. S. a. M. M. Rothenfluh, "Reusable ontologies, knowledge-acquisition tools, and performance systems: PROTÉGÉ-II solution to Sisyphus," *International Journal of Human-Computer Studies,* vol. 2, no. 44, pp. 303-332, 1996.

[49] D. F. R. R. J. a. W. S. McGuinness, “An Environment for Merging and Testing Large Ontologies,” *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference,* 2000.

[50] A. K. KEITA, “Conception Coopérative d’Ontologies Pré-Consensuelles : Application au domaine de l’Urbanisme,” 2007.

[51] Dictionnaire, “Le Petit Robert,” no. ISBN : 978-2-321-00042-6, 2010.

[52] B. Bachimont, “Herméneutique matérielle et Artéfacture : des machines qui pensent aux machines qui donnent à penser,” *Critique du formalisme en intelligence artificielle,* 1996.

[53] J. F. Sowa, “Knowledge Representation: Logical, Philosophical and Computational Foundations,” vol. Pacific Grove, pp. 287-294, 2000.

[54] R. F. T. F. T. G. R. P. T. S. a. W. R. S. Robert Neches, “Enabling Technology For Knowledge Sharing,” vol. 12, no. 3, 1991.

[55] T. R. Gruber, “Toward Principles for the Design of Ontologies Used for Knowledge Sharing,” 1993.

[56] N. Guarino, “Ontologies and Knowledge Bases : Towards a Terminological Clarification,” pp. 25-32, 1995.

[57] A. J. e. T. J. BORST W.N., “Engineering ontologies,” vol. 46, pp. 365-406, 1997.

[58] B. a. P. R. a. K. K. a. R. T. Swartout, “Toward Distributed Use of Large-Scale Ontologies,” pp. 138-148, 1997.

[59] U. M., “Building Ontologies: Towards Unified Methodology,” *16th Annual conference of the British Computer Society ,* 1996.

[60] V. J. I. M. MIZOGUCHI R., “Task Ontology for reuse of problem solving knowledge,” *Towards Very Large Knowledge Bases: Knowledge Building Sharing (KBKS’95),* pp. 46-57, 1995.

[61] A. S. A. B. J. W. I. G. VAN HEUST, “Using explicit ontologies in KBS development,” *Int. J. Human – Computer Studies,* vol. 45, pp. 183-292, 1997.

[62] O. &. D. M. Lassila, “The Role of Frame-based Representation on the Semantic Web,” *Knowledge Systems Laboratory,* 2001.

[63] Y. L., “Roadmap for Tool Support for Collaborative Ontology Engineering,” *Thesis in Computer Science,* p. 119, 2003.

[64] D. E. a. J. Pulido, “Visualizing Ontology Components through Self-Organizing Maps.,” *Proceedings of the Sixth International Conference on Information Visualisation (IV’02),* vol.

95, p. 1093, 2002.

[65] R. G. a. K. Y. Nadine Cullot, " DB2OWL: A Tool for Automatic Database-to-Ontology," 2007.

[66] M. Hepp, "Ontologies: State of the Art, Business Potential, and Grand Challenges," *Ontology Management : Semantic Web, Web Services and Business Applications,* vol. 7, pp. 3-22, 2008.

[67] A. V. Renssen, "Gellish: A Generic Extensible Ontological Language - Design and Application of a Universal Data Structure," Vols. ISBN-13: 978-9040725975, p. 238, 2005.

[68] M. D. S. F. Colace, "A Network Management System Based on Ontology and Slow Intelligence System," *International Journal of Smart Home,* vol. 5, no. 3, pp. 25-38, 2011.

[69] J. M. Panu Korpipää, "An Ontology for Mobile Device Sensor-Based Context Awareness," *Lecture Notes in Computer Science,* vol. 2680, pp. 451-458, 2003.

[70] O. A. a. O. Altan, "An ontology-based visualization for mobile geoinformation services," vol. 6, no. 4, pp. 993-1000, 2011.

[71] M. S. N. S. M. S. M. B. E. K. A. V. Z. L. W. G. O. D. Claudia Villalonga, "Mobile Ontology: Towards a Standardized Semantic Model for the Mobile Domain," *ICSOC 2007 Workshops,* vol. 4907, pp. 248-257, 2007.

[72] M. M. Fox, "Methodology for the Design and Evaluation of Ontologies," 1995.

[73] M. U. M. Gruninger, "Ontologies: Principles, Methods and Applications," vol. 11, no. 2, 1996.

[74] E. Rosch, "Principles of Categorization. Cognition and Categorization," *Hillside, NJ, Lawrence Erlbaum Publishers:,* pp. 27-48, 1978.

[75] K. B. M. M. K. C. J. Matheus, "BaseVISor: A Triples-Based Inference Engine Outfitted to Process RuleML & R-Entailment Rules.," no. W15P7T-05-C-T204, 2006.

[76] P. S. a. M. M. L. Leahu, "Interactionist ai and the promise of ubicomp, or, how to put your box in the world without putting the world in your box," *Proceedings of the 10th international conference on Ubiquitous computing, UbiComp,* pp. 134-143, 2008.

[77] B. J. D. Z. M. Zekri, "On the use of network QoS reputation for vertical handover decision making," *GLOBECOM Workshops (GC Wkshps),* no. ISBN: 978-1-4244-8863-6, pp. 2006-2011, 2010.

[78] B. J. D. Z. M. Zekri, "Context aware vertical handover decision making in heterogeneous wireless networks," *Local Computer Networks (LCN),* pp. 764-768, 2010.

[79] M. Z. B. J. M. Loukil, “A Reputation based Vertical Handover Decision making Framework (R-VHDF),” *to appear in GC'12 Workshop: The 7th IEEE International Workshop on Heterogeneous, Multi-Hop, Wireless and Mobile Networks,* 2012.

[80] J. I. T. M. a. S. B. K. Henricksen, “Middleware for distributed context-aware systems,” *International Symposium on Distributed Objects and Applications (DOA),* vol. 3760, p. 846–863, 2005.

[81] R. F. R. M. R. Guha, Contexts for the Semantic Web, Hiroshima (JP): Proc. 3rd ISWC , 2004, pp. 32-46.

[82] P. S. T. C. S. Y. VRANDECIC D., “The DILIGENT knowledge processes,” *Journal of Knowledge Management,,* vol. 5, no. 9, 2005.

[83] R. Chenhall, “Management control systems design within its organizational context: findings from contingency based research and directions for the future,” vol. 28, no. 2-3, pp. 127-168, 2003.

[84] T. S. a. C. Linnhoff-Popien, “A context modeling survey,” *UbiComp : The Sixth International Conference on Ubiquitous Computing,* 2004.

[85] M. L. B. J. T. Ghariani, “ECO-WISM Energy Consumption Optimization for Wireless Interface Selection Mechanism,” *International Conference on Networking and Future Internet (ICNFI),* 2012.

[86] J. C. S. D. a. D. G. J. Coutaz, “Context is key,” *Communications of the ACM,* vol. 48, no. 3, pp. 49-53, 2005.

[87] B. J. D. Z. M. Loukil, “A two-layered virtualization overlay system using software Avatars,” *Computers and Communications (ISCC),* pp. 1086-1090 , 22-25 juin 2010.

[88] T. G. B. J. D. Z. M. Loukil, “A semantic database framework for context management in heterogeneous wireless networks,” *Wireless and Mobile Computing, Networking and Communications (WiMob),* pp. 35-39, 11-13 Oct. 2010.

Annexes 1 : Manipulation et visualisation de l'ontologie

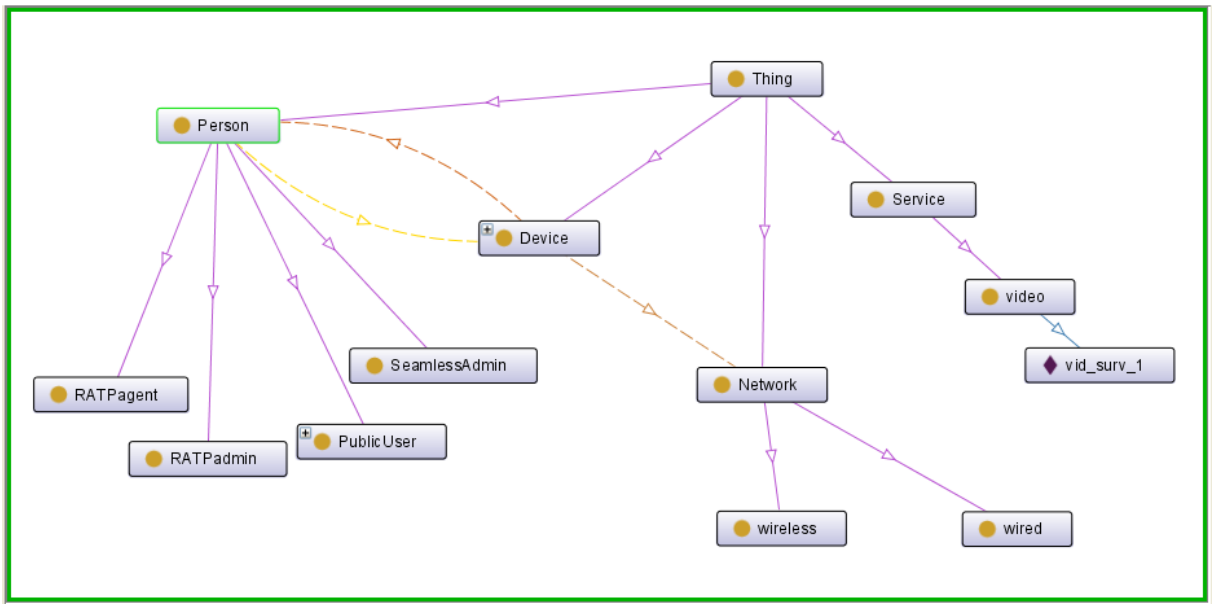


Figure 52 : Exemple d'illustration de relation entre les concepts de l'ontologie

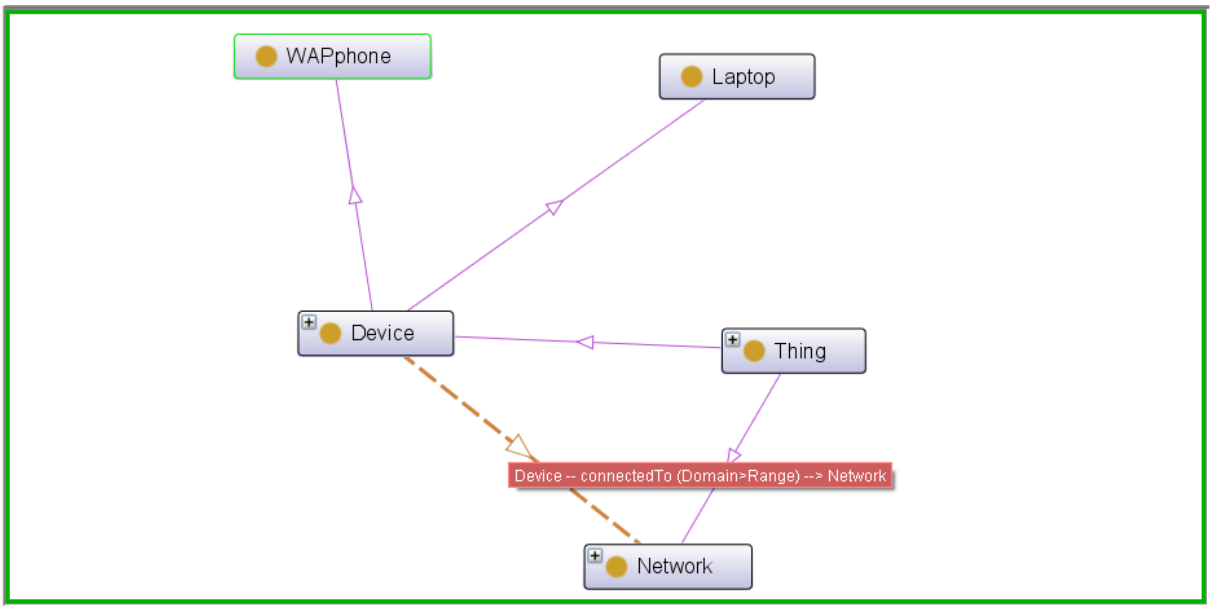


Figure 53 : Exemple d'illustration de relation entre les concepts de l'ontologie (connecté-à)

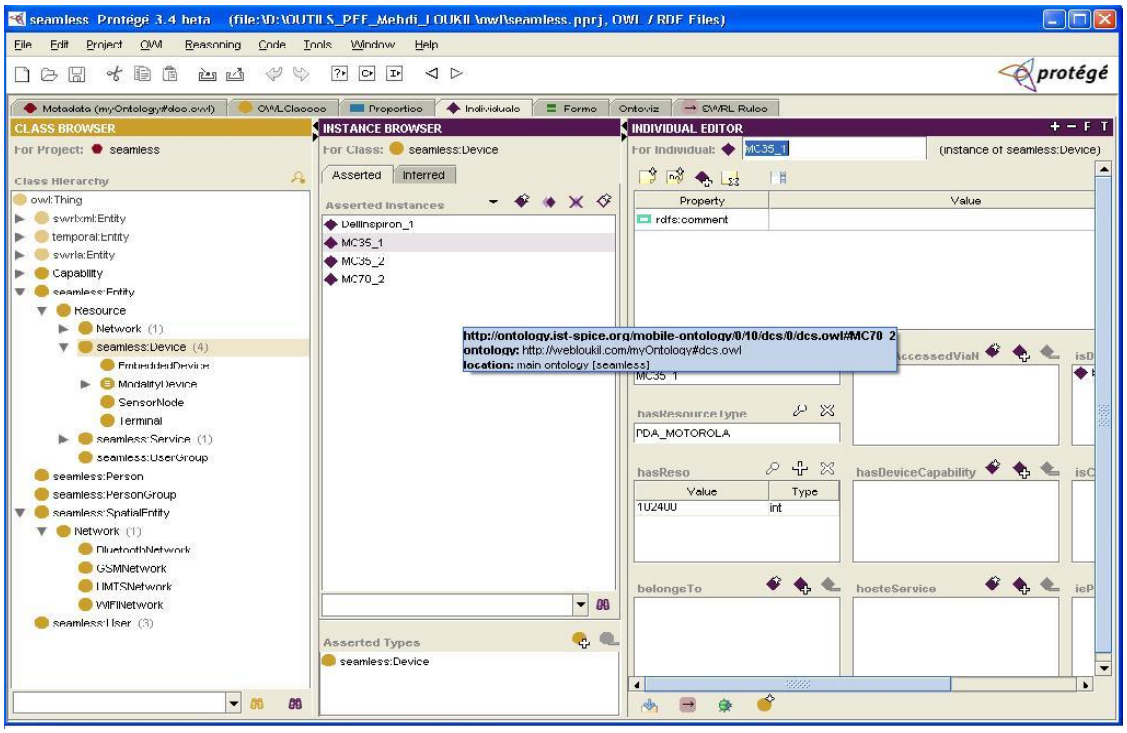


Figure 54 : Une impression d'écran de l'outil Protégé

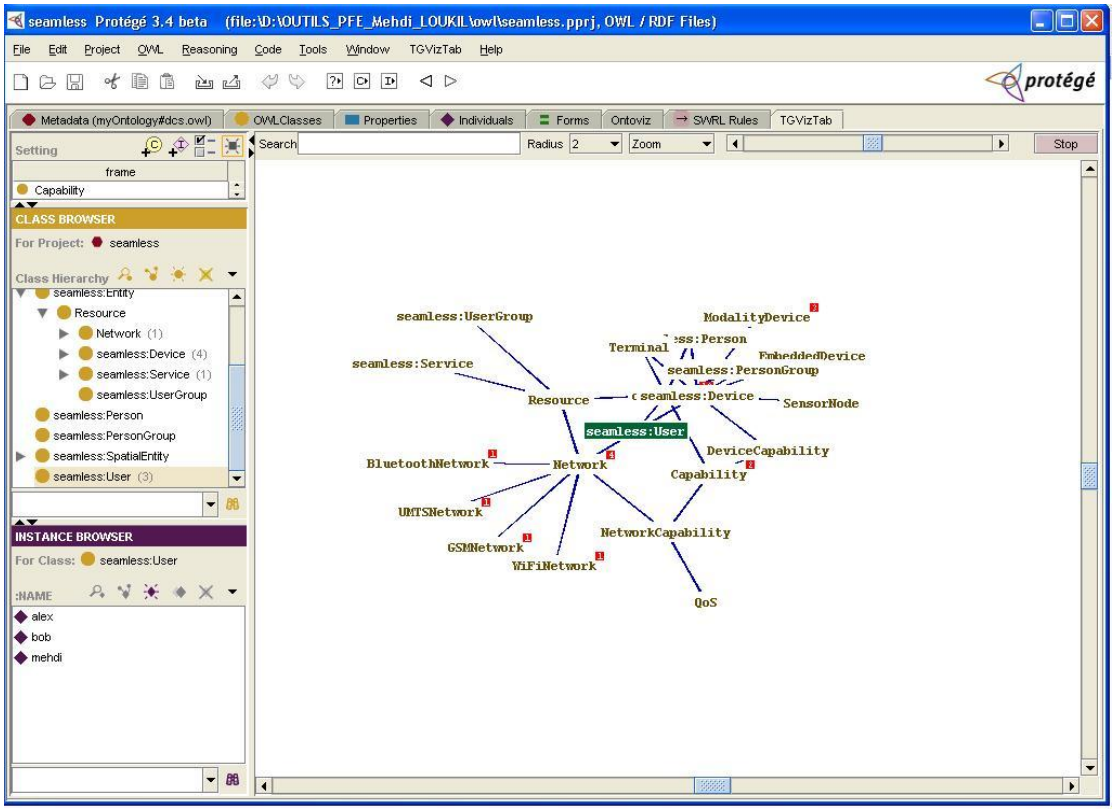


Figure 55 : Un exemple de Plugin Protégé permettant de visualiser graphiquement les concepts

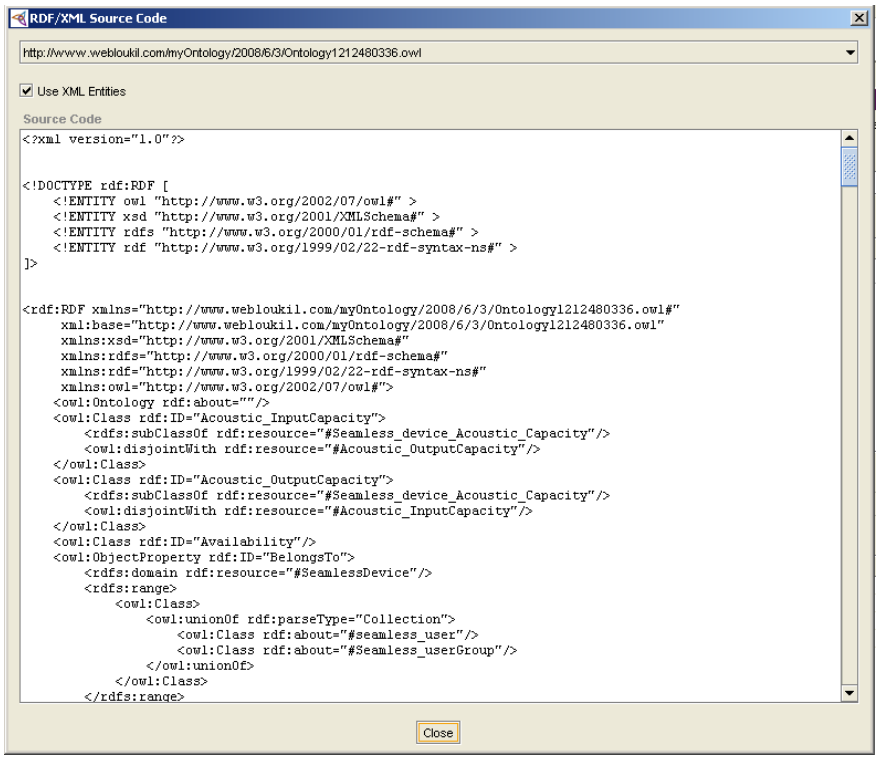


Figure 56 : Extrait du code source OWL/RDF de l'ontologie SEAMLESS

```
  <PublicUser rdf:ID="Bob">
    <name rdf:datatype="string"
    >Bob</name>
    <age rdf:datatype="integer"
    >18</age>
    <address rdf:datatype="string"
    >5 Charles Fourier 91011 Evry France</address>
    <user_id rdf:datatype="integer"
    >10211</user_id>
    <owns>
        <Smartphone rdf:ID="sonyericsson_c902_1">
        <Smartphone rdf:ID="htc_8900_1">
    </owns>
  </PublicUser>

<Smartphone rdf:ID="sonyericsson_c902_1">
 <imei rdf:datatype="numeric"
    >358061003701737</imei>
    <model_name rdf:datatype="string"
    >C902</model_name>
    <resolution_height rdf:datatype="interger"
    >320</resolution_height>
    <resolution_width rdf:datatype="interger"
    >240</resolution_width>
    <mobile_browser_version rdf:datatype="string"
    >NetFront 3.4</mobile_browser_version>
    <pointing_method rdf:datatype="string"
    >joystick</pointing_method>
    <physical_screen_height rdf:datatype="interger"
    >41</physical_screen_height>
    <physical_screen_width rdf:datatype="interger"
    >30</physical_screen_width>
    <max_image_width rdf:datatype="interger"
    >234</max_image_width>
    <seamless_id rdf:datatype="string"
    >Bob_dev_1</seamless_id>
    <ownedBy rdf:resource="#Bob"/>
</Smartphone>
```

```
<Smartphone rdf:ID="htc_8900_1">
 <imei rdf:datatype="numeric"
    >261000031581431</imei>
    <model_name rdf:datatype="string"
    >8900</model_name>
    <resolution_height rdf:datatype="interger"
    >320</resolution_height>
    <resolution_width rdf:datatype="interger"
    >240</resolution_width>
    <mobile_browser_version rdf:datatype="string"
    >7.6</mobile_browser_version>
    <pointing_method rdf:datatype="string"
    >stylus</pointing_method>
    <physical_screen_height rdf:datatype="interger"
    >41</physical_screen_height>
    <physical_screen_width rdf:datatype="interger"
    >30</physical_screen_width>
    <max_image_width rdf:datatype="interger"
    >228</max_image_width>
    <seamless_id rdf:datatype="string"
    >Bob_dev_2</seamless_id>
    <ownedBy rdf:resource="#Bob"/>
</Smartphone>
```

Figure 57 : Illustration des attributs d'une instance du concept Smartphone

```
<video rdf:ID="vid_surv_1">
  <WSDLURI rdf:datatype="string"
  >http://www.seamless.fr/vid_serv/vid_surv_1.wsdl</WSDLURI>
  <URI rdf:datatype="string"
  >157.159.103.114</URI>
  <Description rdf:datatype="string"
  >Video Surveillance. 4 quality classes</URI>
  <VeryHigh rdf:datatype="boolean"
  >true</VeryHigh>
  <High rdf:datatype="boolean"
  >true</High>
  <seamless_id rdf:datatype="string"
  >vid_surv_1</seamless_id>
  <Low rdf:datatype="boolean"
  >true</Low>
  <VeryLow rdf:datatype="boolean"
  >true</VeryLow>
  <min_width_req_resolution rdf:datatype="integer"
  >200</min_width_req_resolution>
  <seamless_id rdf:datatype="string"
  >vid_surv_1</seamless_id>
</video>
```

Figure 58 : Illustration des attributs d'une instance du concept Video (sous classe de service)

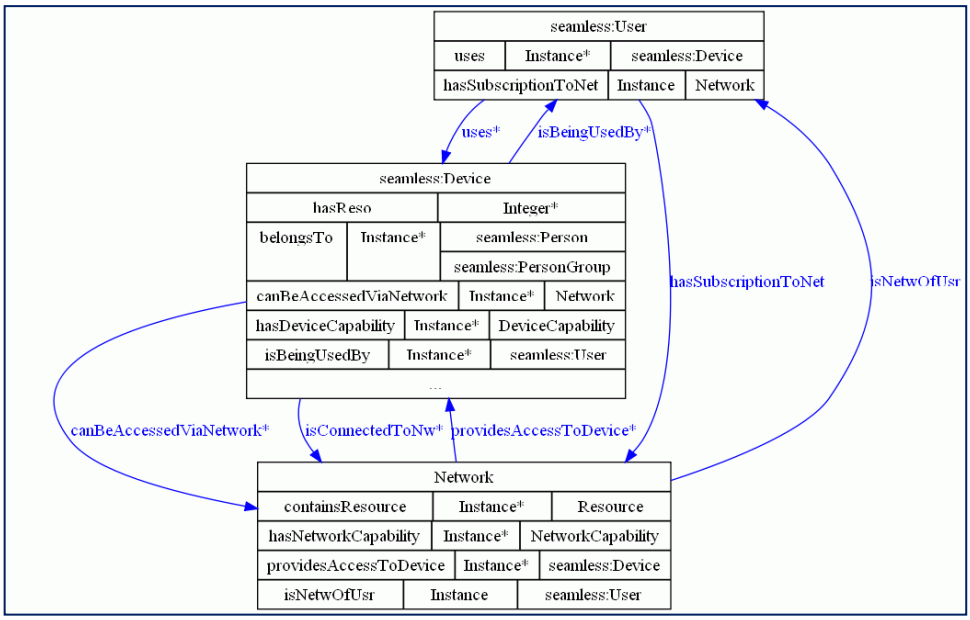
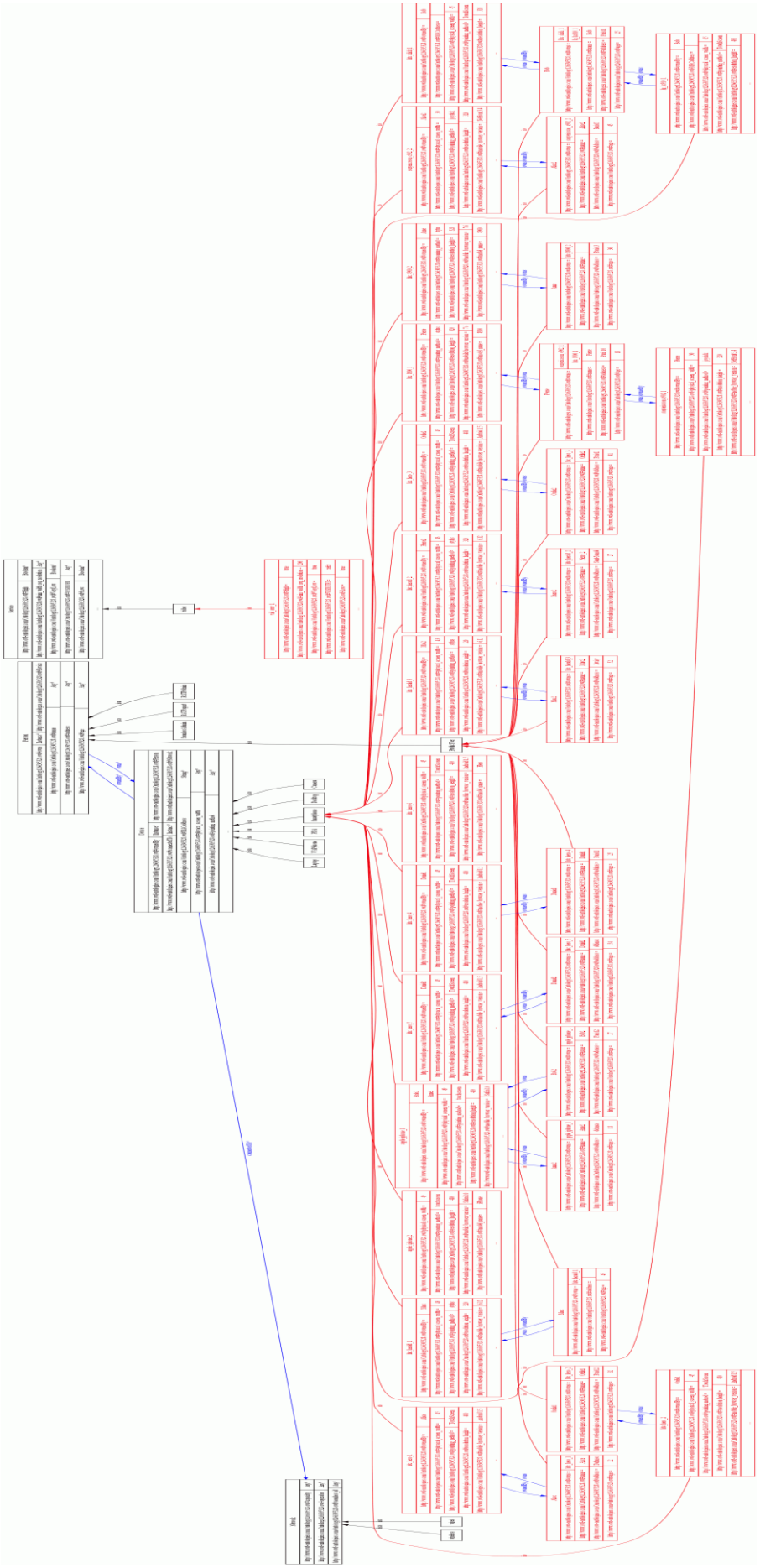
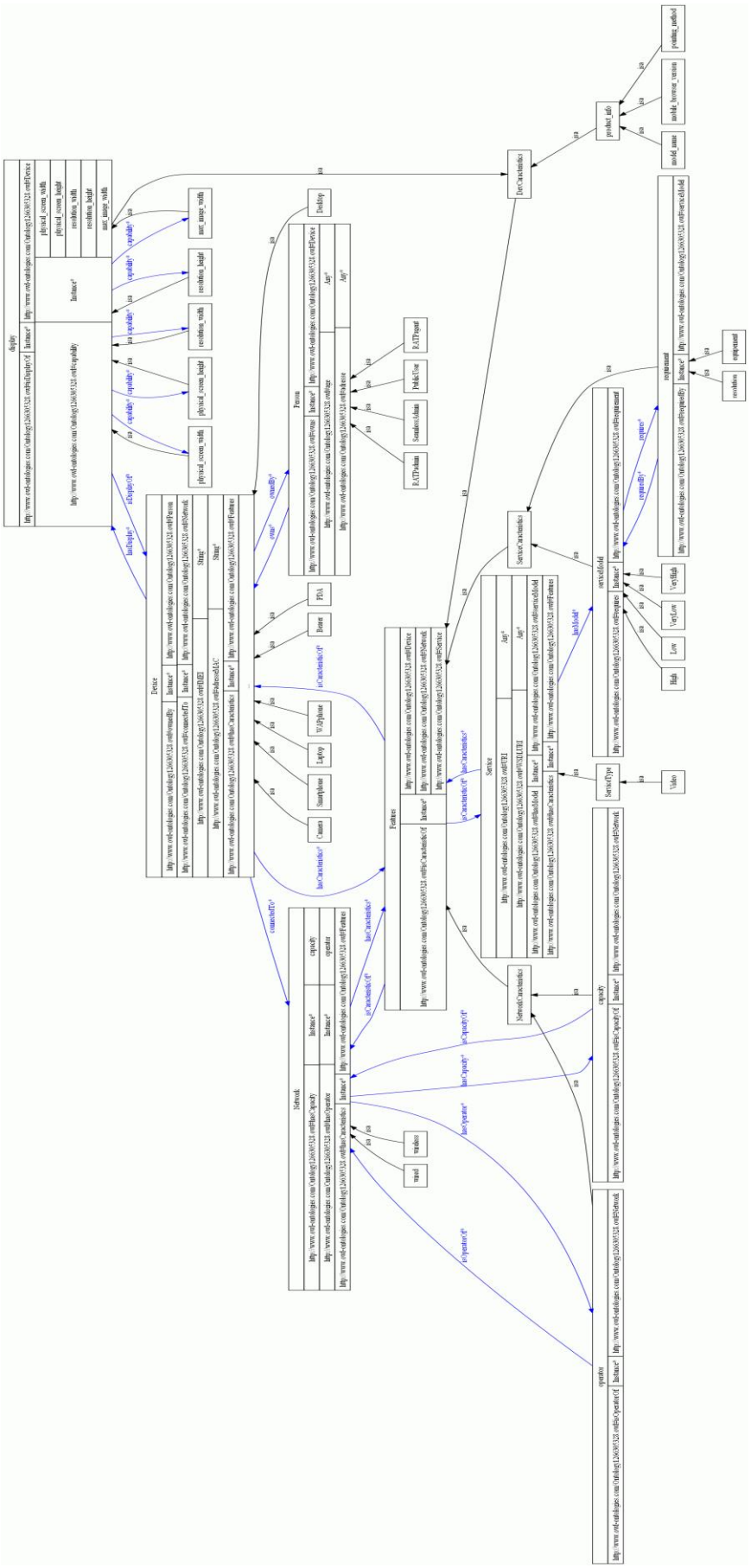
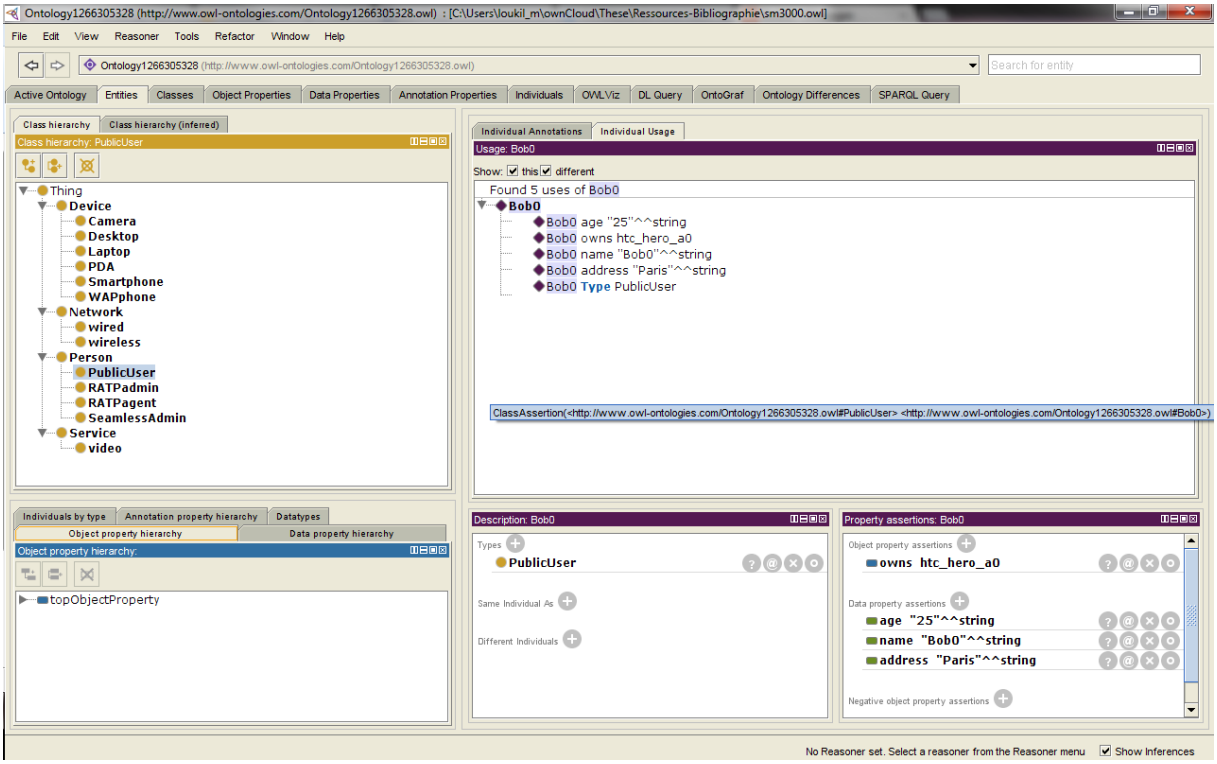
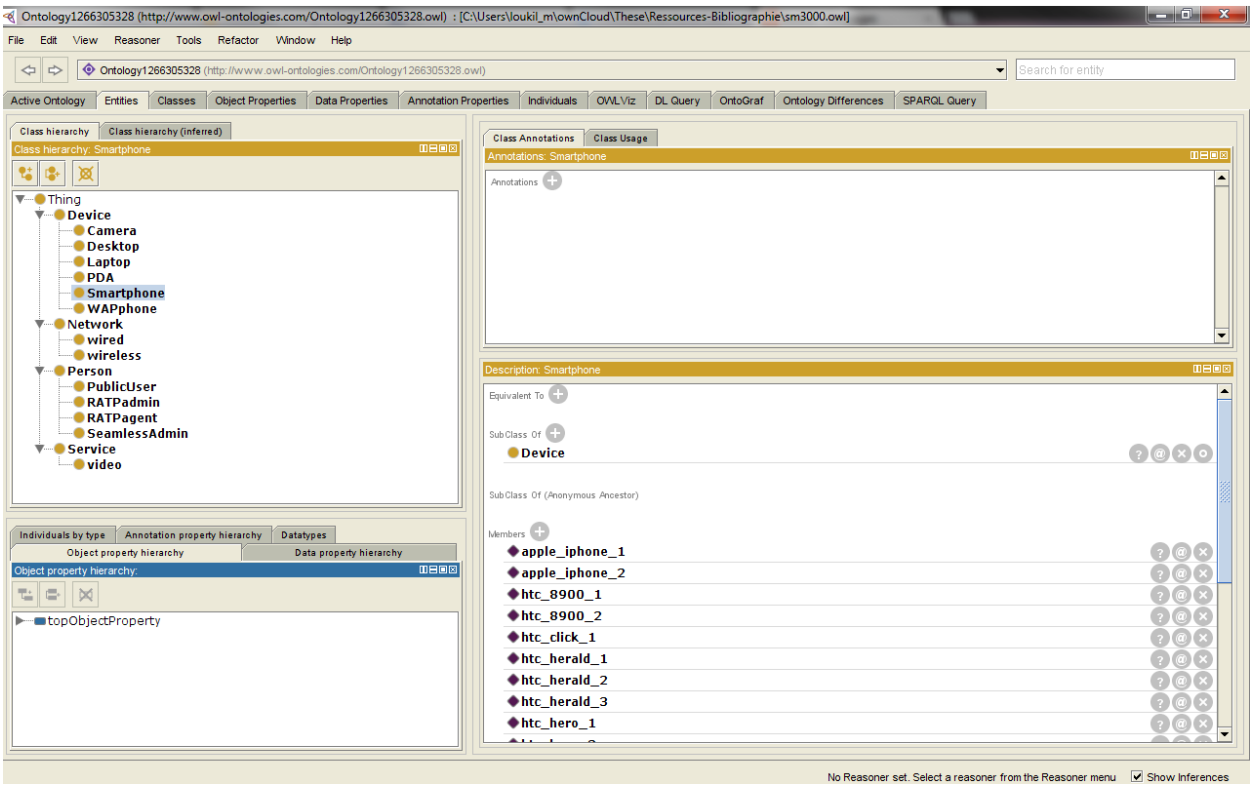


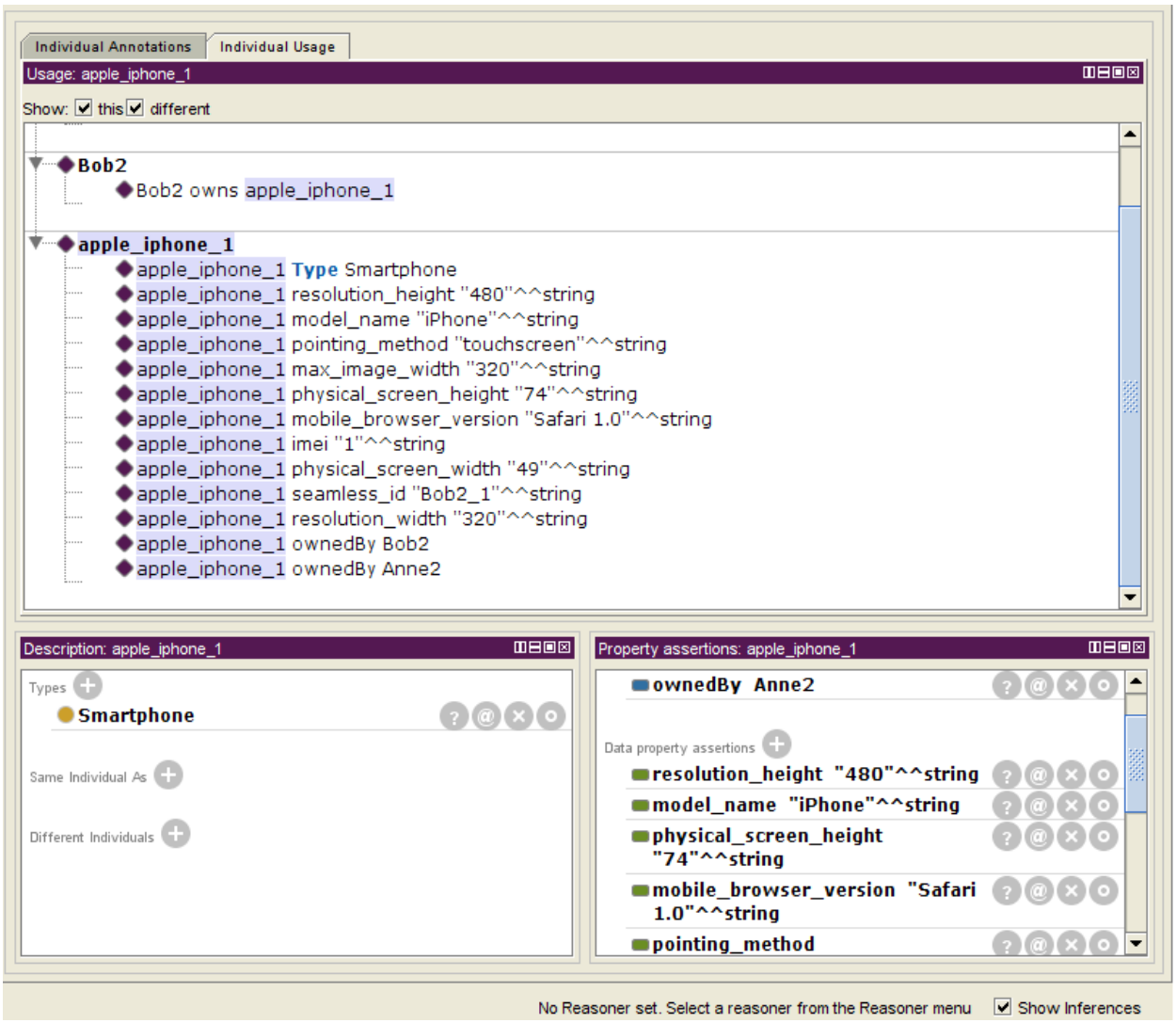
Figure 59 : Exemples de propriétés des concepts utilisateur, terminal et réseau

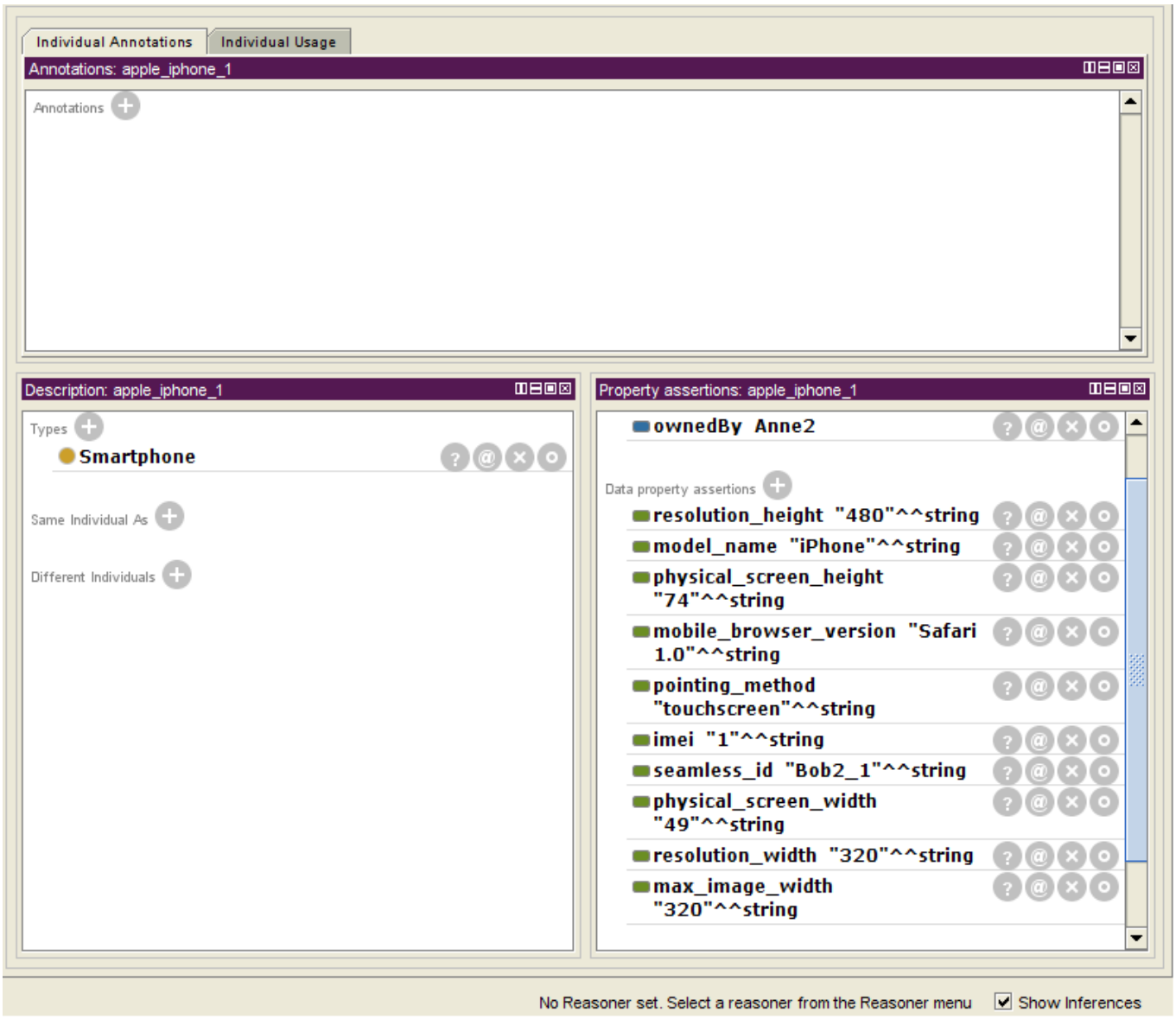


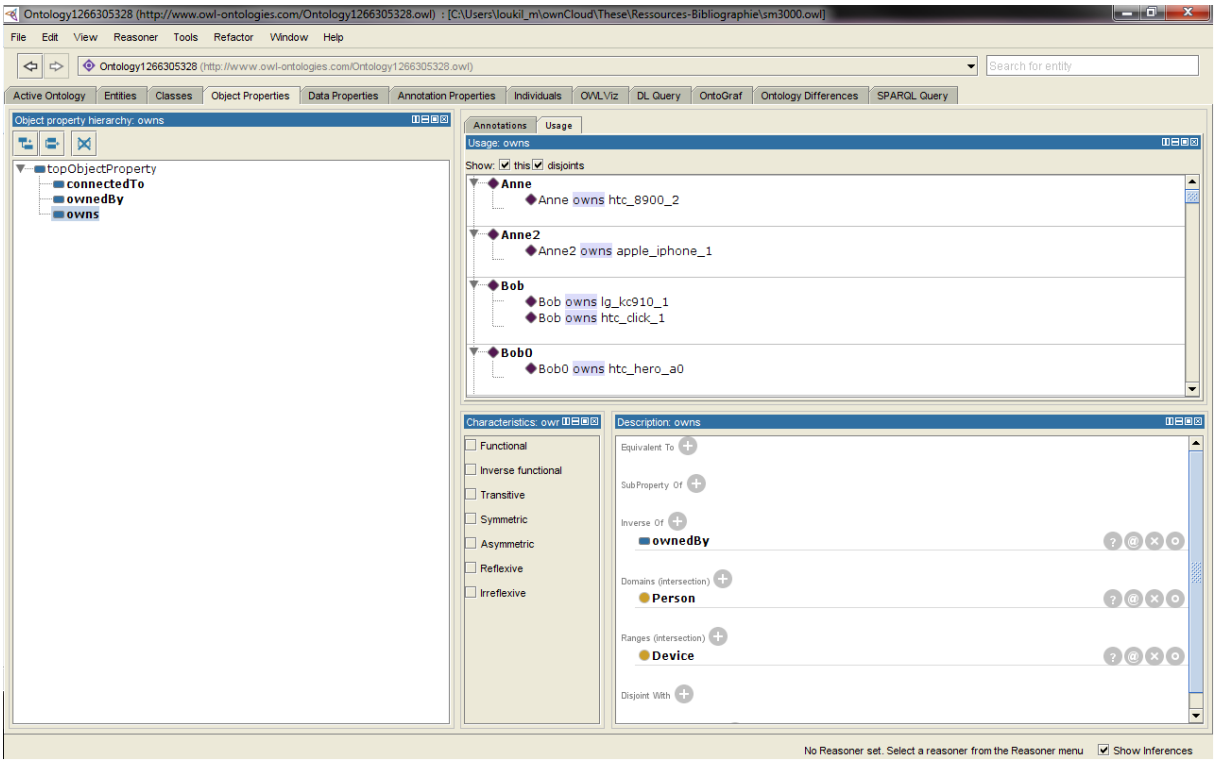


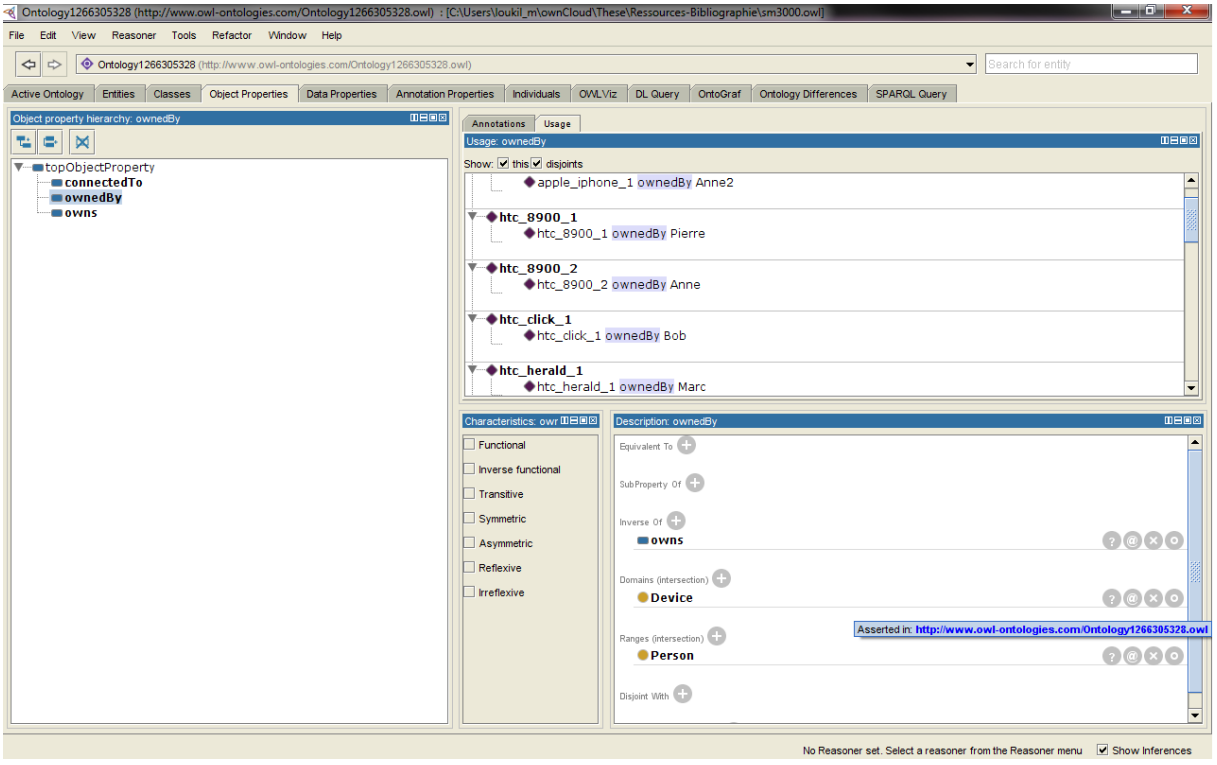

Ontology1266305328 (http://www.owl-ontologies.com/Ontology1266305328.owl) : [C:\Users\loukil_m\ownCloud\These\Ressources-Bibliographie\sm3000.owl]
File Edit View Reasoner Tools Refactor Window Help
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf Ontology Differences SPARQL Query
Class hierarchy: PublicUser
Thing
Device
Camera
Desktop
Laptop
PDA
Smartphone
WAPphone
Network
wired
wireless
Person
PublicUser
RATPadmin
RATPagent
SeamlessAdmin
Service
video
Usage: Bob0
Found 5 uses of Bob0
Bob0
Bob0 age "25"^^string
Bob0 owns htc_hero_a0
Bob0 name "Bob0"^^string
Bob0 address "Paris"^^string
Bob0 Type PublicUser
ClassAssertion(<http://www.owl-ontologies.com/Ontology1266305328.owl#PublicUser> <http://www.owl-ontologies.com/Ontology1266305328.owl#Bob0>)
Object property hierarchy:
topObjectProperty
Description: Bob0
Types
PublicUser
Property assertions: Bob0
owns htc_hero_a0
age "25"^^string
name "Bob0"^^string
address "Paris"^^string
No Reasoner set. Select a reasoner from the Reasoner menu
Show Inferences

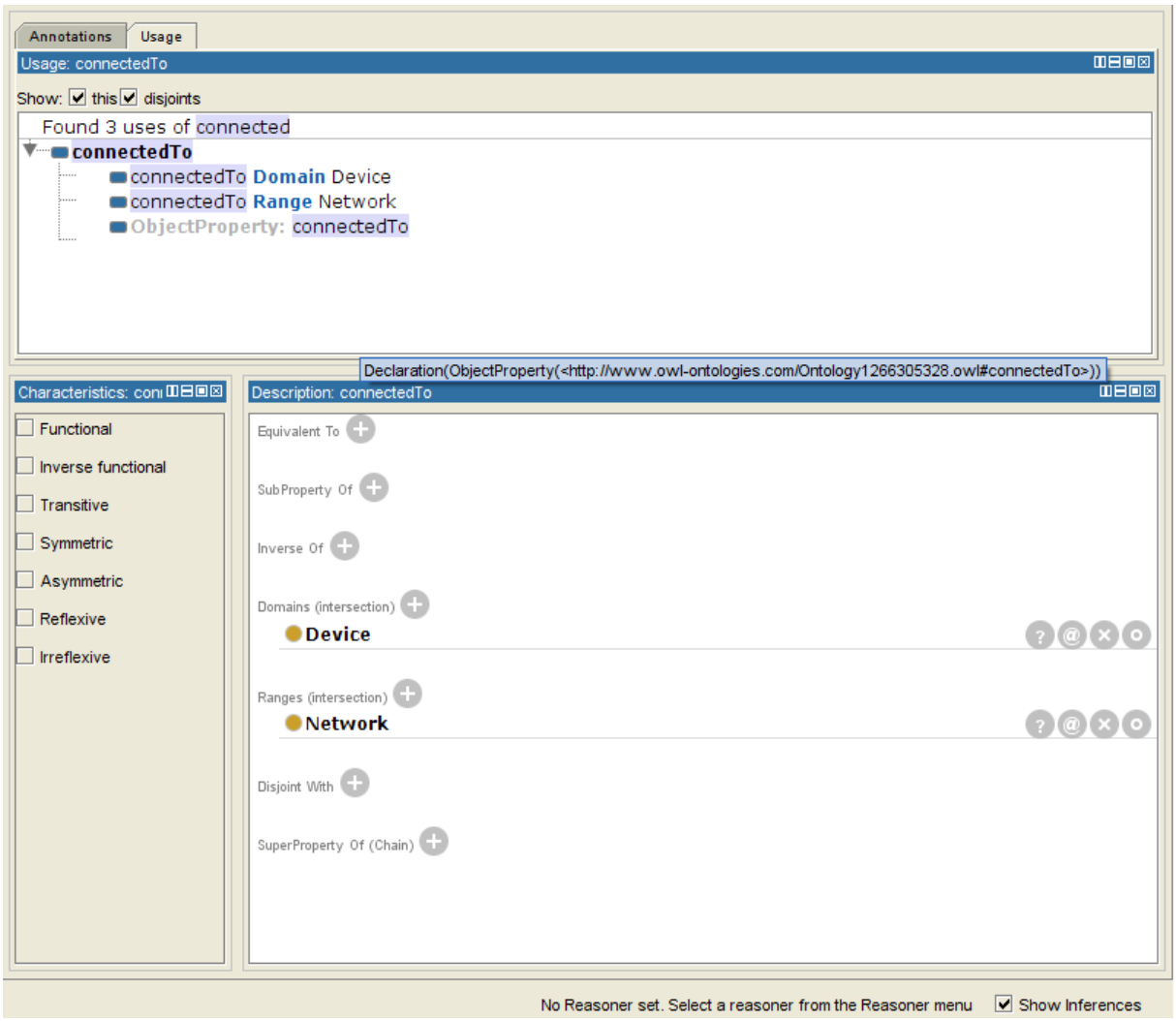

Ontology1266305328 (http://www.owl-ontologies.com/Ontology1266305328.owl) : [C:\Users\loukil_m\ownCloud\These\Ressources-Bibliographie\sm3000.owl]
File Edit View Reasoner Tools Refactor Window Help
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf Ontology Differences SPARQL Query
Class hierarchy: Smartphone
Thing
Device
Camera
Desktop
Laptop
PDA
Smartphone
WAPphone
Network
wired
wireless
Person
PublicUser
RATPadmin
RATPagent
SeamlessAdmin
Service
video
Annotations: Smartphone
Description: Smartphone
Equivalent To
SubClass Of
Device
SubClass Of (Anonymous Ancestor)
Members
apple_iphone_1
apple_iphone_2
htc_8900_1
htc_8900_2
htc_click_1
htc_herald_1
htc_herald_2
htc_herald_3
htc_hero_1
Object property hierarchy:
topObjectProperty
No Reasoner set. Select a reasoner from the Reasoner menu
Show Inferences

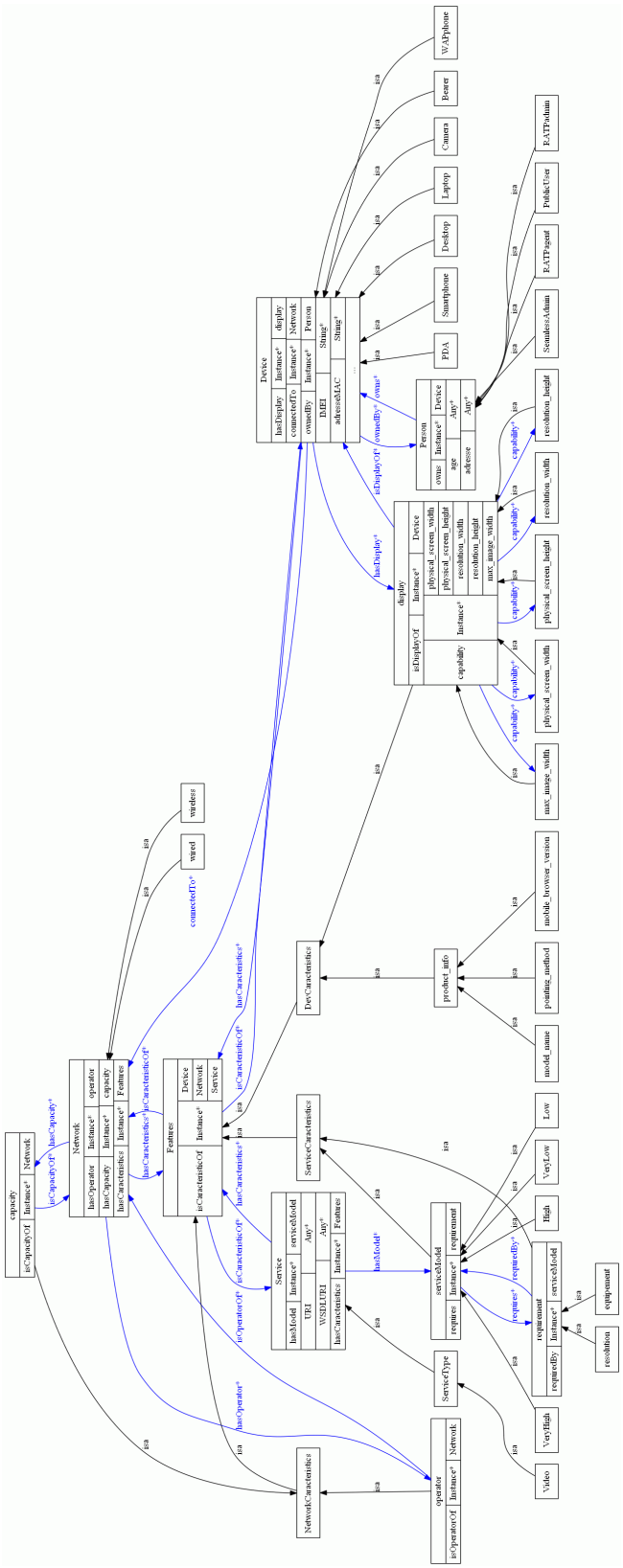

Individual Annotations
Individual Usage
Usage: apple_iphone_1
Show: this different
Bob2
Bob2 owns apple_iphone_1
apple_iphone_1
apple_iphone_1 Type Smartphone
apple_iphone_1 resolution_height "480"^^string
apple_iphone_1 model_name "iPhone"^^string
apple_iphone_1 pointing_method "touchscreen"^^string
apple_iphone_1 max_image_width "320"^^string
apple_iphone_1 physical_screen_height "74"^^string
apple_iphone_1 mobile_browser_version "Safari 1.0"^^string
apple_iphone_1 imei "1"^^string
apple_iphone_1 physical_screen_width "49"^^string
apple_iphone_1 seamless_id "Bob2_1"^^string
apple_iphone_1 resolution_width "320"^^string
apple_iphone_1 ownedBy Bob2
apple_iphone_1 ownedBy Anne2
Description: apple_iphone_1
Types
Smartphone
Same Individual As
Different Individuals
Property assertions: apple_iphone_1
ownedBy Anne2
Data property assertions
resolution_height "480"^^string
model_name "iPhone"^^string
physical_screen_height "74"^^string
mobile_browser_version "Safari 1.0"^^string
pointing_method
No Reasoner set. Select a reasoner from the Reasoner menu
Show Inferences


Individual Annotations
Individual Usage
Annotations: apple_iphone_1
Annotations
Description: apple_iphone_1
Types
Smartphone
Same Individual As
Different Individuals
Property assertions: apple_iphone_1
ownedBy Anne2
Data property assertions
resolution_height "480"^^string
model_name "iPhone"^^string
physical_screen_height "74"^^string
mobile_browser_version "Safari 1.0"^^string
pointing_method "touchscreen"^^string
imei "1"^^string
seamless_id "Bob2_1"^^string
physical_screen_width "49"^^string
resolution_width "320"^^string
max_image_width "320"^^string
No Reasoner set. Select a reasoner from the Reasoner menu
Show Inferences


Ontology1266305328 (http://www.owl-ontologies.com/Ontology1266305328.owl) : [C:\Users\loukil_m\ownCloud\These\Ressources-Bibliographie\sm3000.owl]
File Edit View Reasoner Tools Refactor Window Help
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf Ontology Differences SPARQL Query
Object property hierarchy: owns
topObjectProperty
connectedTo
ownedBy
owns
Usage: owns
Anne owns htc_8900_2
Anne2 owns apple_iphone_1
Bob owns lg_kc910_1
Bob owns htc_click_1
Bob0 owns htc_hero_a0
Characteristics: owns
Functional
Inverse functional
Transitive
Symmetric
Asymmetric
Reflexive
Irreflexive
Description: owns
Equivalent To
SubProperty Of
Inverse Of
ownedBy
Domains (intersection)
Person
Ranges (intersection)
Device
Disjoint With
No Reasoner set. Select a reasoner from the Reasoner menu
Show Inferences


Ontology1266305328 (http://www.owl-ontologies.com/Ontology1266305328.owl) : [C:\Users\loukil_m\ownCloud\These\Ressources-Bibliographie\sm3000.owl]
File Edit View Reasoner Tools Refactor Window Help
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf Ontology Differences SPARQL Query
Object property hierarchy: ownedBy
topObjectProperty
connectedTo
ownedBy
owns
Usage: ownedBy
apple_iphone_1 ownedBy Anne2
htc_8900_1 ownedBy Pierre
htc_8900_2 ownedBy Anne
htc_click_1 ownedBy Bob
htc_herald_1 ownedBy Marc
Characteristics: owns
Functional
Inverse functional
Transitive
Symmetric
Asymmetric
Reflexive
Irreflexive
Description: ownedBy
Equivalent To
SubProperty Of
Inverse Of
owns
Domains (intersection)
Device
Ranges (intersection)
Person
Disjoint With
Asserted in: http://www.owl-ontologies.com/Ontology1266305328.owl
No Reasoner set. Select a reasoner from the Reasoner menu
Show Inferences


Annotations
Usage
Usage: connectedTo
Show: this disjoints
Found 3 uses of connected
connectedTo
connectedTo Domain Device
connectedTo Range Network
ObjectProperty: connectedTo
Declaration(ObjectProperty(<http://www.owl-ontologies.com/Ontology1266305328.owl#connectedTo>))
Characteristics: con
Functional
Inverse functional
Transitive
Symmetric
Asymmetric
Reflexive
Irreflexive
Description: connectedTo
Equivalent To
SubProperty Of
Inverse Of
Domains (intersection)
Device
Ranges (intersection)
Network
Disjoint With
SuperProperty Of (Chain)
No Reasoner set. Select a reasoner from the Reasoner menu
Show Inferences



Annexe 2 : Choix technologiques

Moteur d'inférence

Des inférences peuvent être opérées à partir de faits et d'axiomes contenus par les ontologies en ayant recours à des raisonneurs externes. Parmi ceux-ci, on peut citer les raisonneuses open sources Pellet, Fact ++, F-OWL ou bien les raisonneurs commerciaux RacerPro ou Cerebra. (c.f. deuxième chapitre « Etat de l'art »).

Certains moteurs d'inférence ne peuvent raisonner qu'au niveau terminologique (sur des concepts et des propriétés) alors que des moteurs comme RacerPro et Pellet permettent aussi de réaliser des inférences au niveau des instances.

Pour gérer le contexte des entités physique (terminaux, réseaux, utilisateurs, …) dans le système SEAMLESS, qui représentent des instances des concepts définis dans l'ontologie SEAMLESS, il faut choisir un moteur d'inférence (raisonneur) qui réalise des inférences au niveau des instances. Et entre Pellet et RacerPro nous avons choisi d'utiliser RacerPro. Ce choix se base sur les atouts suivants :

- La documentation sur RacerPro est importante, provenant des concepteurs et des utilisateurs.
- RacerPro permet l'utilisation d'un mécanisme d'abonnement à un concept qui permet d'être informé de la création de nouvelles instances de ce concept.
- RacerPro permet l'ajout d'assertions et d'individus dans les ABox après le chargement de l'ontologie.
- RacerPro permet l'utilisation de règles SWRL.

Par contre les points négatifs de ce raisonneur sont les suivants :

- RacerPro ne permet pas l'utilisation de type de données utilisateur (type défini par l'utilisateur), car il possède ces propres types de données et il effectue une conversion avec les types de base.
- RacerPro est un produit commercial, il n'existe pas de version libre d'utilisation. Cependant il est possible d'obtenir une licence gratuite dans le cadre de la recherche scientifique (c'est avec cette licence que nous avons utilisé RacerPro pour tester son fonctionnement, mais plusieurs fonctionnalités sont désactivées alors nous étions obligé de contourner ces problèmes par le développement d'un simple programme qui ajoute

quelques fonctionnalités nécessaires pour tester l'architecture proposée telle que la fonctionnalité qui permet d'interroger RacerPro à distance).

De l'autre côté, le moteur Pellet a quelques atouts que peut-être ça sera intéressant de prendre en considération pour le choisir dans le cadre du projet ultérieurement. Nous citons entre autres quelques atouts du Pellet :

- Pellet est open-source et développé en Java.
- Pellet est un raisonneur OWL DL complet

Mais les points négatifs de Pellet sont les suivants :

- Pellet possède une documentation pauvre en comparaison à celle de RacerPro. En effet, RacerPro est le plus utilisé et donc le plus documenté par des particuliers.
- Actuellement Pellet ne permet pas l'utilisation de règles SWRL (la prochaine version l'inclura).
- Pellet n'offre pas de système de souscription à un concept.

Par ailleurs, RacerPro est le moteur d'inférence sans doute le plus connu et l'un des plus utilisés dans le domaine pour ces performances et sa stabilité. Racer travaille sur les ontologies modélisées par son langage, mais il accepte des ontologies décrites en RDF ou OWL, ces dernières étant traduites vers le langage utilisé par Racer. Ce moteur d'inférence possède également son propre langage de requête nRQL (new Racerpro Query Language) pour interroger les ontologies sur la ABox et la TBox. Après cette étude comparative nous avons pris la décision de choisir RacerPro pour une première phase car ça reste un produit commercial et « fermé ». En effet, dans une deuxième phase et dans le cadre de l'intégration générale de la plateforme avec les briques logiciels développées par les autres partenaires nous avons utilisé l'API PELLET. Dans la cinquième partie de ce chapitre « implémentation d'une démonstration de test et validation du SCMF » nous allons présenter comment nous avons utilisé ce moteur d'inférence pour implémenter une démonstration afin de tester l'architecture proposée.

Algorithme de mobilité avec la réputation :

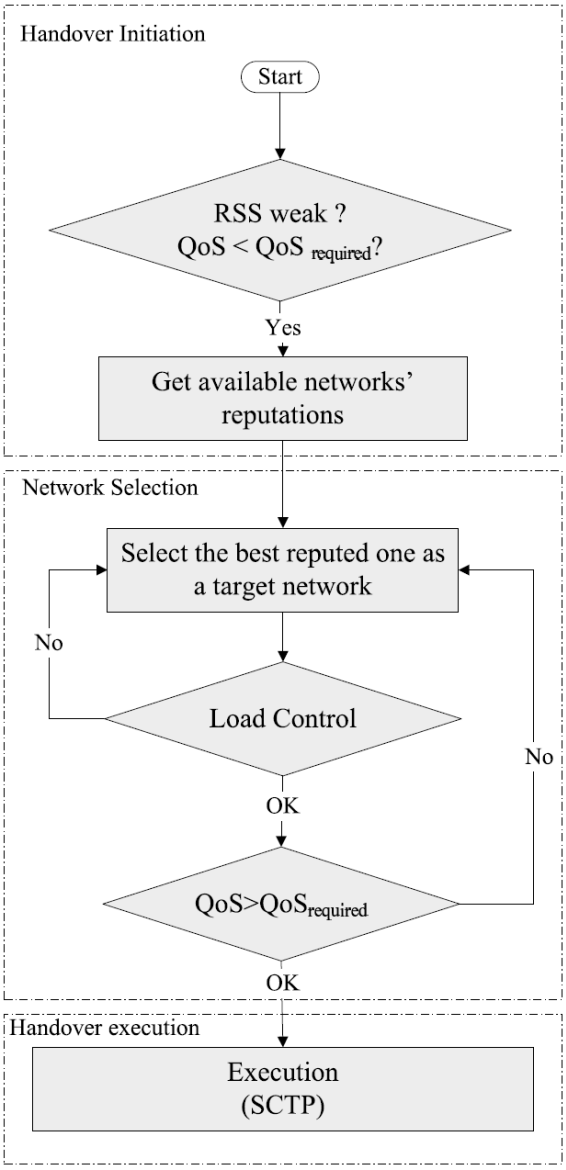


Figure 60 : Algorithme de mobilité avec la réputation

Annexe 3 : Aperçu de l'ontologie Ubiquity-Ont

(Ubiquity-Ont sera publiée et mise à jours sur ce lien : www.mehdi-loukil.com/ubiquity-ont/)

```
<?xml version="1.0" encoding="UTF-8"?>

<tells uri="" xmlns="http://dl.kr.org/dig/2003/02/lang">

   <defconcept name="http://www.owl-
ontologies.com/Ontology1266305328.owl#wired"/>

   <impliesc>

      <catom name="http://www.owl-
ontologies.com/Ontology1266305328.owl#wired"/>

      <catom name="http://www.owl-
ontologies.com/Ontology1266305328.owl#Network"/>

   </impliesc>

   <disjoint>

      <catom name="http://www.owl-
ontologies.com/Ontology1266305328.owl#wired"/>

      <catom name="http://www.owl-
ontologies.com/Ontology1266305328.owl#wireless"/>

   </disjoint>

   <defconcept name="http://www.owl-
ontologies.com/Ontology1266305328.owl#Network"/>

   <defconcept name="http://www.owl-
ontologies.com/Ontology1266305328.owl#wireless"/>

   <impliesc>

      <catom name="http://www.owl-
ontologies.com/Ontology1266305328.owl#wireless"/>

      <catom name="http://www.owl-
ontologies.com/Ontology1266305328.owl#Network"/>

   </impliesc>

   <defconcept name="http://www.owl-
ontologies.com/Ontology1266305328.owl#Service"/>

   <defconcept name="http://www.owl-
ontologies.com/Ontology1266305328.owl#Desktop"/>

   <impliesc>
```

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Desktop"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#agent"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#agent"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Person"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#Person"/>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#PublicUser"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#PublicUser"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Person"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#admin"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#admin"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Person"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#WAPphone"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#WAPphone"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#Smartphone"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Smartphone"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#Laptop"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Laptop"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#wifi"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wifi"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wireless"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#umts"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#umts"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wireless"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#hsdpa"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#hsdpa"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wireless"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#wimax"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wimax"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wireless"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#bluetooth"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#bluetooth"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wireless"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#tetra"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#tetra"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wireless"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#adsl"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#adsl"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wired"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#opticalFibre"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#opticalFibre"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wired"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#email"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#email"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#browsing"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#browsing"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

```
</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#ftp"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#ftp"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#streaming"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#streaming"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#streaming"/>

</impliesc>

<defconcept name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming"/>

<impliesc>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#streaming"/>
```

```
</impliesc>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#MACaddress"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#MACaddress"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</domain>

<rangestring>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#MACaddress"/>

</rangestring>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Network"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryHigh"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryHigh"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

</domain>

<rangeint>
```

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryHigh"/>

</rangeint>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#max_image_width"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#max_image_width"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#physical_screen_width"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#physical_screen_width"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#physical_screen_height"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#physical_screen_height"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</domain>

<rangestring>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

</rangestring>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#min_width_req_resolution"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#min_width_req_resolution"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#age"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#age"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Person"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#pointing_method"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#pointing_method"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

```
</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Network"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#mobile_browser_version"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#mobile_browser_version"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#Low"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#Low"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

</domain>

<rangeint>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#Low"/>

</rangeint>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#imei"/>
```

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#imei"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</domain>

<rangestring>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#imei"/>

</rangestring>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#seamless_id"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#seamless_id"/>

<or>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Network"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

</or>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#WSDLURI"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#WSDLURI"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_width"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_width"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#URI"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#URI"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Service"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#address"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#address"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Person"/>

</domain>

<defattribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_height"/>

<domain>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_height"/>

```
    <catom name="http://www.owl-
ontologies.com/Ontology1266305328.owl#Device"/>
  </domain>
  <defattribute name="http://www.owl-
ontologies.com/Ontology1266305328.owl#High"/>
  <domain>
    <attribute name="http://www.owl-
ontologies.com/Ontology1266305328.owl#High"/>
    <catom name="http://www.owl-
ontologies.com/Ontology1266305328.owl#Service"/>
  </domain>
  <rangeint>
    <attribute name="http://www.owl-
ontologies.com/Ontology1266305328.owl#High"/>
  </rangeint>
  <defattribute name="http://www.owl-
ontologies.com/Ontology1266305328.owl#VeryLow"/>
  <domain>
    <attribute name="http://www.owl-
ontologies.com/Ontology1266305328.owl#VeryLow"/>
    <catom name="http://www.owl-
ontologies.com/Ontology1266305328.owl#Service"/>
  </domain>
  <rangeint>
    <attribute name="http://www.owl-
ontologies.com/Ontology1266305328.owl#VeryLow"/>
  </rangeint>
  <defattribute name="http://www.owl-
ontologies.com/Ontology1266305328.owl#name"/>
  <domain>
```

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#name"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Person"/>

</domain>

<defrole name="http://www.owl-ontologies.com/Ontology1266305328.owl#connectedTo"/>

<domain>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#connectedTo"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>

</domain>

<range>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#connectedTo"/>

<or>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Network"/>

</or>

</range>

<defrole name="http://www.owl-ontologies.com/Ontology1266305328.owl#owns"/>

<domain>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#owns"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Person"/>

</domain>

<range>

```
      <ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#owns"/>
      <or>
        <catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>
      </or>
    </range>
    <equalr>
      <ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#owns"/>
      <inverse>
        <ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#ownedBy"/>
      </inverse>
    </equalr>
    <defrole name="http://www.owl-ontologies.com/Ontology1266305328.owl#ownedBy"/>
    <domain>
      <ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#ownedBy"/>
      <catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Device"/>
    </domain>
    <range>
      <ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#ownedBy"/>
      <or>
        <catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Person"/>
      </or>
```

```
</range>
<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#orange333"/>
<instanceof>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#orange333"/>
<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#hsdpa"/>
</instanceof>
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#orange333"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>
<sval>90</sval>
</value>
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#orange333"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>
<sval>orange</sval>
</value>
<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#stephane"/>
<instanceof>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#stephane"/>
<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#PublicUser"/>
```

```
</instanceof>
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#stephane"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#age"/>
<sval/>
</value>
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#stephane"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#address"/>
<sval/>
</value>
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#stephane"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#name"/>
<sval/>
</value>
<related>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#stephane"/>
<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#owns"/>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Desktop_3"/>
</related>
```

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#brayen"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#brayen"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#PublicUser"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#brayen"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#age"/>

<sval>40</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#brayen"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#address"/>

<sval>nice</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#brayen"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#name"/>

<sval>brayen</sval>

</value>

<related>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#brayen"/>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#owns"/>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

</related>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#MACaddress"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#max_image_width"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#physical_screen_height"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#pointing_method"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#mobile_browser_version"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#imei"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_width"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_height"/>

<sval>200</sval>

</value>

<related>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#ownedBy"/>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bob"/>

</related>

<definidividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Laptop_4"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Laptop_4"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Laptop"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Laptop_4"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#MACaddress"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Laptop_4"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

<sval/>

</value>

<related>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Laptop_4"/>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#ownedBy"/>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#guy"/>

</related>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bob"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bob"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#PublicUser"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bob"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#age"/>

<sval>45</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bob"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#address"/>

<sval>paris</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bob"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#name"/>

<sval>bob nomBob</sval>

</value>

<related>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bob"/>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#owns"/>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#PDA_5"/>

</related>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Smartphone"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#max_image_width"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#physical_screen_height"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#pointing_method"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#mobile_browser_version"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#imei"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_width"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_height"/>

<sval/>

</value>

<related>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#ownedBy"/>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#alice"/>

</related>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#guy"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#guy"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#PublicUser"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#guy"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#age"/>

<sval>50</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#guy"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#address"/>

<sval>lyon</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#guy"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#name"/>

<sval>guy nomGuy</sval>

</value>

<related>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#guy"/>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#owns"/>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Laptop_4"/>

</related>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#orange22"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#orange22"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#umts"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#orange22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>

<sval>70</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#orange22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>

<sval>orange</sval>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#sfr222"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#sfr222"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#umts"/>

</instanceof>

```
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#sfr222"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>
<sval>40</sval>
</value>
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#sfr222"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>
<sval>sfr</sval>
</value>
<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#free222"/>
<instanceof>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#free222"/>
<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#hsdpa"/>
</instanceof>
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#free222"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>
<sval>22</sval>
</value>
```

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#free222"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>

<sval>free</sval>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#alice"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#alice"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#PublicUser"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#alice"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#age"/>

<sval>25</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#alice"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#address"/>

<sval>evry</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#alice"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#name"/>

<sval>Alice nomAlice</sval>

</value>

<related>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#alice"/>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#owns"/>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#iphone12"/>

</related>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#sfr444"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#sfr444"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#hsdpa"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#sfr444"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#sfr444"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>

<sval/>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryHigh"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#min_width_req_resolution"/>

<sval>1024</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#Low"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#seamless_id"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#WSDLURI"/>

<sval>Ontology/1266305328.owl#videoStreaming_22</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#URI"/>

<sval>http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#High"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryLow"/>

<ival>0</ival>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming_25"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming_25"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming_25"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryHigh"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming_25"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#Low"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming_25"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#URI"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming_25"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#High"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#audioStreaming_25"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryLow"/>

<ival>1</ival>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryHigh"/>

<ival>0</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#min_width_req_resolution"/>

<sval>200</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#Low"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#WSDLURI"/>

<sval>/Ontology1266305328.owl#videoStreaming_24</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#URI"/>

<sval>http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#High"/>

<ival>0</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_24"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryLow"/>

<ival>1</ival>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Smartphone"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#MACaddress"/>

<sval>000023427A22</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#max_image_width"/>

<sval>400</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#physical_screen_height"/>

<sval>300</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

<sval>htc_hero</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#pointing_method"/>

<sval>tactil</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#mobile_browser_version"/>

<sval>2.5</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#imei"/>

<sval>32169546513213169</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_width"/>

<sval>400</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_height"/>

<sval>300</sval>

</value>

<related>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#htc11"/>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#ownedBy"/>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#brayen"/>

</related>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_23"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_23"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_23"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryHigh"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_23"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#min_width_req_resolution"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_23"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#Low"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_23"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#WSDLURI"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_23"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#URI"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_23"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#High"/>

<ival>1</ival>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#videoStreaming_23"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#VeryLow"/>

<ival>0</ival>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_1"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_1"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_1"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

<sval>sony</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_1"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_width"/>

<sval>1024</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_1"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_height"/>

<sval>800</sval>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#free1"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#free1"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#umts"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#free1"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>

<sval>45</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#free1"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>

<sval>free</sval>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_2"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_2"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_2"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_2"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_width"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Camera_2"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#resolution_height"/>

<sval/>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap1"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap1"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wifi"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap1"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>

<sval>30</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap1"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>

<sval>orange</sval>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap2"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap2"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wifi"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap2"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>

<sval>20</sval>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap2"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>

<sval>free</sval>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap3"/>

```
<instanceof>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap3"/>
<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#wifi"/>
</instanceof>
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap3"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>
<sval/>
</value>
<value>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#ap3"/>
<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>
<sval/>
</value>
<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bt22"/>
<instanceof>
<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bt22"/>
<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#umts"/>
</instanceof>
<value>
```

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bt22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bt22"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#operator"/>

<sval/>

</value>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Desktop_3"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Desktop_3"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#Desktop"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Desktop_3"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#MACaddress"/>

<sval/>

</value>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Desktop_3"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#model_name"/>

<sval/>

</value>

<related>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#Desktop_3"/>

<ratom name="http://www.owl-ontologies.com/Ontology1266305328.owl#ownedBy"/>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#stephane"/>

</related>

<defindividual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bt222"/>

<instanceof>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bt222"/>

<catom name="http://www.owl-ontologies.com/Ontology1266305328.owl#hsdpa"/>

</instanceof>

<value>

<individual name="http://www.owl-ontologies.com/Ontology1266305328.owl#bt222"/>

<attribute name="http://www.owl-ontologies.com/Ontology1266305328.owl#capacity"/>

<sval>55</sval>

</value>

```
   <value>

      <individual name="http://www.owl-
ontologies.com/Ontology1266305328.owl#bt222"/>

      <attribute name="http://www.owl-
ontologies.com/Ontology1266305328.owl#operator"/>

      <sval>bt</sval>

   </value>

</tells>
```